VŠB-TECHNICKÁ UNIVERZITA OSTRAVA

EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Webová aplikace pro implementaci montážního deníku

Web Application for Construction Diary Implementation

Student: **Mikula Radim**

Vedoucí bakalářské práce: **RNDr. Martiník Ivo, Ph.D.**

Ostrava 2013

Místopřísežně prohlašuji, že jsem celou práci, včetně většiny příloh, zpracoval samostatně.

V Havířově, dne 3. května 2013

.....

Radim Mikula

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce RNDr. Ivu Martiníkovi, Ph.D. nejen za poskytnutí veškerých informací a cenných rad, ale také za čas strávený společnými konzultacemi a velkou ochotu, díky čemuž mohla vzniknout tato bakalářská práce.

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra aplikované informatiky

Zadání bakalářské práce

Student: **Radim Mikula**

Studijní program: B6209 Systémové inženýrství a informatika

Studijní obor: 6209R001 Aplikovaná informatika

Téma: Webová aplikace pro implementaci montážního deníku
Web Application for Construction Diary Implementation

Zásady pro vypracování:

1. Úvod
2. Teoretická východiska práce v oblastech technologií Java a MySql
3. Analýza a popis stávajícího systému
4. Návrh koncepce řešení pro vývoj webové aplikace
5. Zhodnocení navržené koncepce a její implementace na portálu firmy
6. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků bakalářské práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

HALL, Marty. *Java: servlety a stránky JSP*. Praha: Neocortex, 2001. ISBN 80-863-3006-0.
KOFLER, Michael. *Mistrovství v MySQL*. 5. vyd. Brno: Computer Press, 2007.
ISBN 978-80-251-1502-2.
SCHILDT, Herbert. *Java 7: výukový kurz*. Brno: Computer Press, 2012. ISBN 978-80-251-3748-2.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **RNDr. Ivo Martiník, Ph.D.**

Datum zadání: 23.11.2012
Datum odevzdání: 10.05.2013

VYSOKÁ ŠKOLA BÁŇS. - TECHNICKÁ UNIVERZITA Ekonomická fakulta 1 OSTRAVA

Ing. Petr Rozehnal, Ph.D.
vedoucí katedry

prof. Dr. Ing. Dana Dluhošová
děkanka fakulty

Obsah

1 Úvod

V roce 1992 organizace National Science Foundation, která v té době spravovala páteřní síť Internetu, umožnila připojení komerčních subjektů. Díky tomuto a také zdokonalování osobních počítačů, se Internet začal rozvíjet do podoby, kterou známe dnes.

Internet je globální síť, která umožňuje sdílení libovolných informací celému světu. Kromě komunikačních prostředků (např. emaily, faxy) je její hlavní výhodou možnost prezentovat prakticky cokoliv. Pod pojmem prezentovat se rozumí poskytnutí základních statických informací jako je název a sídlo firmy, ale také možnost dynamicky generovaných informací. Tyto složitější způsoby dynamického poskytování informací jsou obvykle realizovány v podobě infomačních systémů. Tento systém na základně vstupních informací uživatele poskytuje zpracované výstupní data.

Pro zpracování bakalářské práce jsem zvolil téma „Webová aplikace pro implementaci montážního deníku“, protože mne zajímají moderní programovací jazyky a v programování jsem si našel zálibu. V této práci využívám programovací jazyk Java, neboť jde o přední jazyk Internetu. V dnešní době být webovým vývojářem, který programuje webové aplikace, znamená ovládat jazyk Java, a mým snem je stát se webovým vývojářem. Java je navíc součástí revoluce v prostředí chytrých telefonů, protože se používá při programování pro operační systém Android.

Cílem této práce je analýza, návrh konceptuálního modelu a realizace webové aplikace pro implementaci Montážního deníku zpracovávající data o zaměstnancích soukromé firmy. Při realizaci této práce je kladen především důraz na jednoduchost použití uživatelské aplikace. Realizace funkčního systému je prováděna za použití moderního vývojového prostředí Netbeans IDE 7.3 s využitím technologie Java Enterprise Edition 6 a relační open source databází MySQL.

Navrhovaný informační systém Montážního deníku je určen pro společnost Elektro Heinich (viz http://www.heinich.com). Tato firma se zabývá veškerou elektroinstalací, montáží antén a satelitů, elektro-revizemi, montáží topných systémů, osvětlením včetně návrhů atd. Firma vznikla v roce 1992 a působí především v Moravskoslezském kraji. Elektronický montážní deník se může uplatnit i v jiných odvětvích. Jakými jsou firmy zabývající veškerou montáží, označované jako dílčí práce na stavbě. To může být např. sklenářství, svářečství. Jednoduše řečeno veškeré firmy, které poskytují řemesla.

Hlavním cílem práce je tedy vytvořit informační systém v prostředí webové služby protokolu HTTP, který umí komunikovat s databázovým systémem. V dnešní době se jedná o nejrozšířenější typ informačních systémů. Zkráceně se nazývají dynamicky generované webovské stránky na základě přístupu k databázi. Hlavní cíl práce je rozdělen do několika podcílů:

1. Seznámení s problematikou technologií JSP, Java Servletů, relačních databází a jejich využití v infomačních systémech.
2. Analýza stávajícího informačního systému ve firmě.
3. Návrh konceptuálního modelu pro nově vytvářený informační systém a jeho implementace do relačního modelu.
4. Realizace webového rozhraní Montážního deníku s využitím technologie Java Enterprise Edition 6.

Vzhledem rozsahu popisovaných technologií není v této práci prostor vysvětlit podrobně všechny výše uvedené pojmy.

Tato práce se skládá ze dvou částí, z části teoretické a praktické. Teoretická část a zároveň druhá kapitola pojednává o teoretických východiscích nutných pro návrh, realizaci a implementaci systému Montážní deník. V této kapitole jsou popsány technologie Java Servletů a JSP stránek. Dále popisuje základní vlastnosti databáze, objekty databáze a databázové modely. Kapitola slouží jako příprava k praktické části.

Praktická část je rozdělena do kapitol (3., 4. a 5.). Třetí kapitola popisuje analýzu stávajícího systému firmy. Čtvrtá kapitola se zabývá vytvořením datové struktury pomocí tříúrovňové koncepce a dále popisuje realizaci webového rozhraní pomocí Java Enterprise Edition 6. Pátá kapitola slouží pro zhodnocení navrženého systému Montážního deníku.

2 Teoretická východiska práce v oblastech technologií Java a MySql

Tato kapitola se zabývá teoretickými východisky, které jsou nutné pro návrh, realizaci a implementaci informačního systému Montážního deníku. Pro grafický návrh je použita notace jazyků CSS a HTML, pro její realizaci a implementaci objektově-orientovaný jazyk Java, konkrétně platforma Java EE 6. Uložení konkrétních dat bude provedeno s využitím relační databáze MySQL.

2.1 Programovací jazyk Java

Počítačový programovací jazyk Java patří mezi nejdůležitější a nejrozšířenější programovací jazyky na světě, protože jde o přední jazyk Internetu. V dnešní době být profesionálním webovým vývojářem aplikací znamená umět ovládat jazyk Java, protože programy v tomto jazyce jsou přenositelné, tudíž nezávislé na platformě. (SCHILDT, 2012)

Jak tvrdí (Cadenhead, 2011) počítačový program, zvaný také software, je způsob říct počítači, co má konkrétně dělat. Veškerá činnost počítače od načítání operačního sytému až po vypínání, je prováděno pomocí programu. Počítačové hry jsou také programy, operační systém je program i ovladač pro tiskárnu je program. Můžeme tedy říct, že program je konečná statická sekvence instrukcí procesoru.

2.1.1 Historie Javy

Firma Sun Microsystems od roku 1991 vyvíjela programovací jazyk na principech C a C++ pro běžná elektronická zařízení ovládaná zabudovaným mikroprocesorem, jako je topinkovač, mikrovlnná trouba a dálkové ovládání. Zakladatelé Javy jsou James Gosling, Patrick Naughton, Chris Warth, Ed Frank a Mike Sheridan. Původní název byl tohoto programovacího jazyka byl „Oak“, ale v roce 1995 byl přejmenován na Java. V roce 1993 si firma Sun Microsystems uvědomila důležitost vzrůstající celosvětové webové sítě (Web). V květnu roku 1995 firma Sun Microsystems představila Javu oficiálně na konferenci SunWorld v San Franciscu. Během této přednášky bylo zřejmé, že se jedná o programovací jazyk, který bude hrát významnou úlohu při programování webových aplikací. (Herout, 2010)

2.1.2 Základní rysy Javy

Podle (SCHILDT, 2012) mezi základní rysy patří bezpečnost a přenositelnost. Pro bezpečnost je důležité zabránit přístupu škodlivým kódům. Za takový škodlivý kód se

považuje např. trojský kůň. Příkladem trojského koně může být program pojmenovaný waterfalls.src, který po spuštění otevírá porty počítače a poskytuje crackerům[1] vzdálený přístup do počítače uživatele. Proto je nutné zabránit v provádění škodlivého kódu.

Přenositelnost je hlavním hlediskem v celosvětové webové síti, protože k ní je připojena řada různých typů počítačů a operačních systémů. Mechanismus pro zajištění bezpečnosti a přenositelnosti se nazývá bajtový kód.

2.1.2.1 Bajtový kód

Kompilátor je software, který překládá program do jazyka, který je spustitelný nad daným operačním systémem. Výstupem kompilátoru Javy není spustitelný kód, ale bajtový kód (bytecode). Jak tvrdí (SCHILDT, 2012, str. 26). „*Bajtový kód je vysoce optimalizovaná sada instrukcí navržených provádění běhovým systémem Javy, který se nazývá virtuální procesor Javy zvaný také JVM (Java Virtual Machine).“* Tím se jazyk Java liší od ostatních programovacích jazyků, které jsou navržené pro kompilování do spustitelného kódu, proto je lze spustit jen na přesně daném operačním systému a hardwaru. Díky skutečnosti, že program v Javě provádí JVM, pomáhá řešit problémy s webovými programy.

Program v Javě lze spustit v jakémkoliv prostředí, neboť v tomto prostředí stačí pouze implementovat JVM a umístit nad příslušný hardware a operační systém, softwarovou vrstvu JVM, která simuluje univerzální procesor disponující jednotnou sadou instrukcí a nad tímto JVM je spustitelný program v jazyce Java. Je to nejsnazší způsob vytváření přenositelných programů.

Skutečnost, že program provádí JVM, může programu zamezit v generování škodlivých efektů mimo systém. Dále je bezpečnost vylepšena určitými omezeními. (SCHILDT, 2012)

Jak bylo zmíněno, dosahuje se přenositelnosti pomocí bajtového kódu. Jeho interpretace je úkolem speciálních programů, nazývaným společně Java platforma.

2.1.2.2 Java platforma

Java platforma se skládá ze dvou částí. První část tvoří JVM, který zajišťuje vazbu na hardware a interpretující bajtový kód. Druhou částí Java platformy tvoří Java Core API. (Herout, 2010)

[1] Cracker je označení pro člověka, který zneužívá své vědomosti o počítačové bezpečnosti při průnicích do softwaru.

2.1.2.3 Java Core API

Jak uvádí (Herout, 2010) ve své publikaci „*Učebnice Jazyka Java*“, zkratka API znamená *Application Programming Interface* (aplikační programové rozhraní). Pod touto zkratkou je obsaženo značné množství veřejných knihoven tříd, které jsou považovány za standardní. To znamená, že když program využívá metody z API, není kód součástí programu, protože je součástí API. Prakticky to znamená, že programy obsahují pouze kód, který jsme využili této práci a soubory, ve kterých jsou naše přeložené programy uloženy. Tudíž mají poměrně malou velikost.

2.1.3 Základní vlastnosti jazyka Java

Důležitou roli hrají i další faktory, které jsou popsány v seznamu základních vlastností Javy, které uvádí (SCHILDT, 2012):

- **Jednoduchý –** Obsahuje kompletní sadu rysů, které usnadňují jeho osvojení a používání.
- **Bezpečný** – Bezpečné prostředky pro tvorbu internetových aplikací.
- **Přenositelný –** Programy lze provádět v libovolném prostředí, kde existuje JVM.
- **Objektově orientovaný –** Ztělesňuje moderní objektově-orientovanou filozofii programování.
- **Robustní –** Podporuje bezchybné programování, protože kontrola je prováděna za běhu.
- **Vícevláknové programování –** Integrovaná podpora pro vícevláknové programování.
- **Architektonicky neutrální –** Jazyk Java není svázán určitým operačním systémem.
- **Interpretovaný –** Podporuje kód nezávislý na platformě.
- **Vysoce výkonný –** Bajtový kód je optimalizován na rychlost provádění.
- **Distribuovaný –** Jazyk je navržen pro distribuované prostředí Internetu.
- **Dynamický –** Programy nesou značné množství běhových informací o typech, které se za běhu používají pro ověřování a vyhodnocování přístupu k objektům.

2.1.3.1 Vícevláknové programování

I když Java obsahuje řadu inovativních prvků, patří mezi nejdůležitější vestavěná podpora pro vícevláknové programování (multithreaded programming). Vícevláknový program obsahuje dvě nebo více částí programu, které mohou běžet souběžně. Každá část takového programu se nazývá vlákno (thread) a každé vlákno definuje svou vlastní část provádění procesu. Proces je množina prostředků používaných při specifické části daného programu. Jazyk Java umožňuje práci s programovými vlákny na úrovni definice jazyka, a proto podporuje filozofii programových vláken. Vícevláknové provádění je speciální formou multitaskingu, což je souběžné provádění více úloh. Úloha neboli proces je konečná množina prostředků používaných při vykonávání specifické instance daného programu. (SCHILDT, 2012)

Dle (SCHILDT, 2012) je výhodou vícevláknového provádění, že nám umožňuje psát efektivnější programy tím, že nám dovoluje využívat dobu nečinnosti procesoru, která je typická pro většinu programů. Díky vícevláknovému provádění může program během této doby nečinnosti provádět jinou úlohu. Např. zatímco jedna část programu odesílá soubor přes Internet, může jiná část číst vstup z klávesnice a další může nahrávat nějaká data pro odeslání, jak uvádí (Herout, 2010).

Během posledních pár let se staly víceprocesorové a vícejádrové systémy samozřejmostí, ale musíme podotknout, že i jednoprocesorové systémy jsou nadále rozšířené. Je proto nutné pochopit, že vícevláknové prvky Javy fungují na obou typech systémů. Avšak v jednojádrovém systému sdílejí prováděná vlákna procesor, přičemž každé vlákno obdrží určitý podíl procesorového času. Proto neběží dvě vlákna současně, ale využívají dobu nečinnosti procesoru. Nicméně ve víceprocesorových či vícejádrových systémech je možné, aby dvě vlákna prováděla svou činnost současně. To může vylepšit efektivitu programu a rychlost určitých operací. (SCHILDT, 2012)

Vlákna v Javě mohou být na sobě téměř zcela nezávislá. Takový program ovšem nemá větší smysl. Obvykle vlákna spolu nějakým způsobem spolupracují a čím větší je jejich spolupráce, tím je větší nutnost zajistit správnou interakci. K tomu se používá synchronizace. (Herout, 2010)

Synchronizace

Při používání více vláken je nezbytné zabezpečit jejich koordinaci. Prostředek, při kterém se dosahuje takové koordinace, se nazývá synchronizace. Například když jedno vlákno

zapisuje do jistého souboru, je nutné zabezpečit, aby jiné programovací vlákno nezapisovalo také do téhož souboru.

Klíčem k synchronizaci v Javě je koncepce monitoru objektu. Monitor funguje tak, že implementuje koncepci zámku. Když jedno vlákno uzamkne objekt, žádné jiné vlákno k němu nemůže získat přístup. Po ukončení činnosti tohoto vlákna se objekt odemkne a je k dispozici pro použití jiného vlákna. (SCHILDT, 2012)

Důvod studia problematiky programových vláken je důležitý pro programování webových aplikací. Protože uživatelé přistupují pomocí protokolu HTTP k serverovým komponentám a jejich požadavky jsou zpracovány právě programovým vláknem.

2.1.4 Objektově orientované programování

Všechny programy v Javě jsou objektově-orientované. Objektově-orientované programování vzniklo ze strukturovaného programování a kombinací nových konceptů. Výsledkem je odlišný způsob uspořádání programu. Hlavním principem jsou data řídící přístup ke kódu. Definujeme data a rutiny, které mohou nad těmito daty pracovat. Datový typ specifikuje, jaký druh operací mohou data aplikovat.

V objektově-orientovaném jazyce je základním kamenem třída (class), proto je třída jádrem jazyka Java. Třída představuje soubor proměněných, konstant a podprogramů. Proměnným se říká datové položky, ve kterých je uložen stav objektu. Podprogramy jsou metody, které umožňují manipulaci s proměnnými, a tudíž mění stav objektu, a také popisují schopnosti objektu.

Podle (HALL, 2001) třída je šablona, která definuje tvar objektu a nemá přidělenou žádnou paměť. Objekt je datový prvek, který je vytvořen podle vzoru třídy. Objekty jsou instance třídy. Třída je skupina plánu, jenž stanovuje sestavu objektu. Podle jednoho vzoru třídy lze vytvořit libovolné množství objektů, které mají stejné schopnosti, také mohou mít navzájem různé nebo stejné vztahy, které můžeme nazvat jako hodnoty datových položek.

Všechny objektově orientované jazyky mají společné rysy (SCHILDT, 2012):

- zapouzdření,
- polymorfizmus,
- dědičnost.

2.1.4.1 Zapouzdření

Zapouzdření je mechanismus, který umožňuje možnost deklarace datového typu třída. Dále udržuje kód i data v bezpečí před narušením a nesprávným použitím. Jedná se o spojení dat a metod dané konstrukcí třídy. Data se vždy vyskytují společně s metodami, které s nimi manipulují. Důsledkem zapouzdření je autorizovaný přístup k datům, kdy zajistíme, aby s daty nebylo možno manipulovat z vnějšku třídy jinak, než pomocí metod této třídy. (Herout, 2010)

2.1.4.2 Polymorfizmus

Polymorfizmus, českým slovem vícetvarost nebo mnohotvarost, je vlastnost, která dovoluje manipulovat s prvky příbuzného předem neznámého typu. Teorie ohledně polymorfismu je značně rozsáhlá, my se zde zaměříme pouze na uvedené konstatování.

2.1.4.3 Dědičnost

Díky dědičnosti můžeme vytvořit obecnou třídu, která definuje vlastnosti společné nějaké skupině souvisejících prvků. Z této třídy pak mohou dědit další, specifičtější třídy. Z nichž každá přidá věci, které jsou pro ni jedinečné. Jestliže má každá třída maximálně jednu nadtřídu, hovoříme o jednoduché dědičnosti. (SCHILDT, 2012)

(SCHILDT, 2012, str. 256) uvádí, že „*Java podporuje dědičnost tím, že umožňuje jedné třídě začlenit do své deklarace jinou třídu*.“

2.1.5 Vývojové nástroje

Pro spouštění a kompilování programů je nutné mít na svém počítači nainstalovanou sadu Java Development Kit (JDK). JDK nabízí zdarma společnost Oracle. V dnešní době je dostupná verze JDK 7.

Sada JDK běží v prostředí příkazového řádku. Nejedná se o okénkovou aplikaci, ani o integrované vývojové prostředí (IDE).

Kromě příkazového řádku dodávaného se sadou JDK existuje pro Javu několik vysoce kvalitních prostředí IDE: NetBeans a Eclipse. Tato vývojová prostředí nabízejí mnohem lepší formu pro psaní Java kódu, než je příkazový řádek. Prostředí je velice užitečné při vyvíjení a nasazování komerčních aplikací. Dále obsahuje grafické uživatelské rozhraní. Vše pracuje na doplňujícím JDK, které běží za pozadí vývojového prostředí. Proto je nutné mít oba nástroje v operačním systému, pokud vyvíjíme Java programy. (SCHILDT, 2012)

Všechny programy pro práci Javou jsou dostupné zdarma na http://java.sun.com. Z praktického hlediska se vývojové nástroje rozdělily do tří skupin, konkrétně tedy Java SE (Standard Edition), Java ME (Micro Editon) a Java EE (Enterpise Edition). V této práci se věnujeme především Java EE 6, která je určena k vývoji podnikových informačních systémů a programování serverových aplikací, na rozdíl od Java SE, která slouží pro programy na straně klienta. (Herout, 2010)

2.2 Komponenty distribuovaných aplikací

Vývojáři dnes stále více uvědomují potřebu distribuované, transakční, a přenosné aplikace, které budou mít vliv na rychlost, bezpečnost a spolehlivost na straně serveru. Enterprise aplikace poskytují obchodní logiku podniku. Jsou řízeny centrálně a často integrované s jiným podnikovým softwarem. Ve světě informačních technologií, musí být podnikové aplikace navrženy, vytvářeny a vyrobené za co nejmenší náklady s větší rychlostí. (Jendrock, Evans, Gollapudi, Haase, & Srivathsa, 2010)

S Java Platform Enterprise Edition (Java EE) je vývoj podnikových aplikací snadnější a rychlejší. Hlavním cílem Java EE platformy je poskytnout vývojářům výkonnou sadu rozhraní API, které vedou ke zkrácení doby vývoje, snížení komplexnosti a zlepšení výkonu aplikací. (Heffelfinger, 2011)

V roce 2009 byla představena finální verze Java EE 6 společností Oracle, která přinesla mnoho změn. Mezi hlavní novinky patří dependency injection, díky kterému dochází ke sjednocení zápisu napříč frameworky, nebo bean validation, které jsou standardizovány mezi vrstvami aplikace. Dále byly výrazně rozšířeny funkčnosti EJB komponent a Java Servletů. (Jendrock, Evans, Gollapudi, Haase, & Srivathsa, 2010)

2.2.1 Java Enterpise Edition 6

Java Enterpise aplikace se skládá z mnoha komponent a několika kontejnerů. Každá komponenta je nasazena do určitého Java EE kontejneru. Komponenta má svou dílčí vlastnost a funkci, můžeme si ji představit jako část systému. Tato komponenta je dále vložena do kontextu Enterpise aplikace s ostatními komponentami systému. Kontejner slouží jako rozhraní pro komunikaci mezi komponentou a okolím. Výhoda tohoto řešení spočívá ve znovupoužitelnosti komponent a snadné rozšiřitelnosti systému. O veškeré komponenty

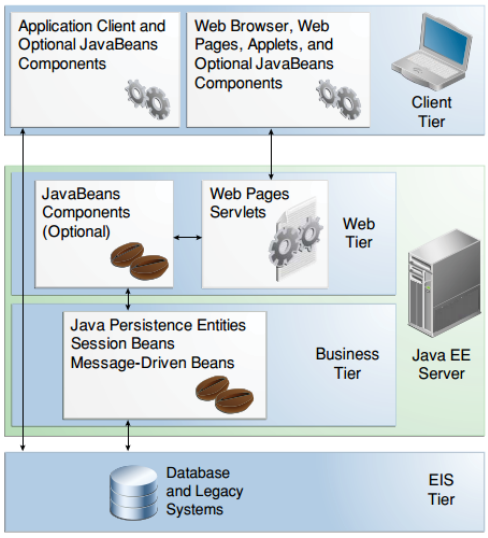
vložené do kontejneru se stará Java EE aplikační server. Tudíž se stará o veškeré životní cykly těchto komponent. (Jendrock, Evans, Gollapudi, Haase, & Srivathsa, 2010)

Java EE aplikační server obsluhuje několik požadavků ve stejném čase a k obsluze každého požadavku používá jedno programovací vlákno.

Specifikace Java EE říká, že každý aplikační server musí obsahovat 2 kontejnery (Basham, Sierra, & Bates, 2008):

1. webový kontejner,
2. Enterprise Java Bean kontejner.

Komponenty mohou být využívány na straně klienta (např. Java Applet) i na straně serveru (např. Java Servlet).



Obrázek 2-1 Architektura enterpise aplikací [2]

2.2.2 Klientská vrstva

Klientská vrstva obsahuje 3 druhy komponent:

1. webový klient,
2. Applet,
3. aplikační klient.

[2] Zdroj: *The Java EE 6 Tuturial: Basic Concepts fourth Edition, 2010.*

2.2.2.1 Webový klient

Webovým klientem se rozumí webový prohlížeč jako je např. Mozilla Firefox. Webový prohlížeč je software, který ví, jak má komunikovat s web serverem. Hlavním úkolem webového prohlížeče je interpretovat HTML kód a předložit webovou stránku uživateli. (Basham, Sierra, & Bates, 2008)

HTML

Jak uvádí (Basham, Sierra, & Bates, 2008), pro vývoj webových stránek se používá HTML, který popisuje jak má web stránka vypadat a jaké má mít funkce. HyperText Markup Language, označovaný zkratkou HTML je hlavním prostředkem pro vytváření webových stránek, který umožňuje publikaci dokumentů na Internetu. Je charakterizován množinou značek (tzv. tagů) a jejich atributů.

2.2.2.2 Aplikační klient

Aplikační klient je zpravidla desktopová aplikace, kterou si můžeme představit jako běžné programy známé z jiných programovacích jazyků. (Herout, 2010)

2.2.3 Webová vrstva

Tato vrstva obsahuje komponenty, kterými mohou být:

1. Java Servlety,
2. JSP stránky,
3. JSF stránky.

2.2.4 Java Servlety

Základní programová komponenta, která je obecně určena ke generování multimediálních dynamických webových stránek. K jejímu provozu potřebujeme webový kontejner Java EE aplikačního serveru, Java Servlet není samostatně spustitelný program. Při tvorbě Java Servletu musíme dodržovat přesné specifikace, aby se zachovala přenositelnost komponent na různé druhy Java EE aplikačních serverů. K Java Servletu se přistupuje standardně pomocí protokolu HTTP a tato programová komponenta je napsaná v jazyce Java, která je potomkem HttpServlet a obsluhuje požadavky (request) a odpovědi (response). (Basham, Sierra, & Bates, 2008)

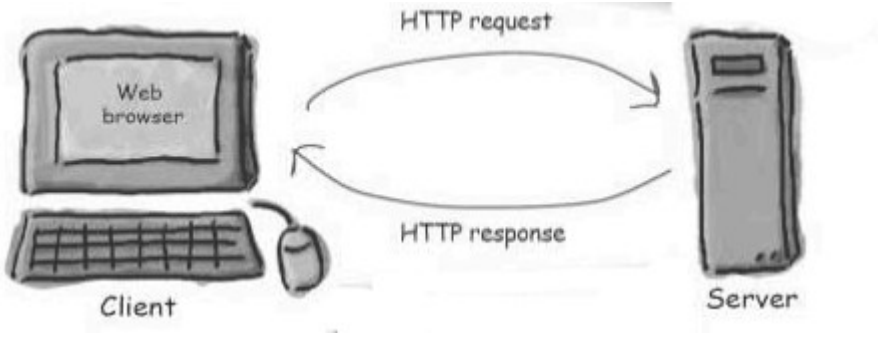
2.2.4.1 Základní struktura Java Servletu

Jak uvádí (HALL, 2001), obvyklým typem požadavku webového prohlížeče pro webové stránky je požadavek typu GET a patří k metodám protokolu HTTP. Prohlížeč vytváří tento požadavek, jestliže uživatel na adresním řádku webového prohlížeče napíše URL (Uniform Resoucre Locator) adresu. Java Servlety rovněž mohou velmi snadno manipulovat s požadavky typu POST, které se vytvářejí v případě odeslání HTML formuláře.

Jak již bylo řečeno, abychom získali Java Servlet, musíme deklarovat třídu, která je podtřídou třídy HttpServlet a předepsat metody doGet nebo doPost v závislosti na tom, zda data jsou odesílána pomocí metody požadavků GET nebo POST. (HALL, 2001)

HTTP protokol

Dle (Basham, Sierra, & Bates, 2008) HTTP běží na vrcholu protokolu TCP/IP. TCP (Transmission Control Protokol) zajišťuje bezpečný přenos souboru z jedné sítě do druhé i když je soubor rozdělen do paketů při odesílání. IP (Internet Protokol) slouží pro odesílání souboru optimální trasou sítě. HTTP protokol je další síťový protokol, který je součástí sady protokolů TCP/IP a slouží k získání kompletního požadavku a odpovědi z jednoho zařízení na druhé. Struktura HTTP komunikace je jednoduchá požadavek (request)/odpověď (response). Webový prohlížeč odesílá požadavek a server odpověď.



Obrázek 2-2 Komunikace protokolu HTTP [3]

2.2.4.2 Životní cyklus Java Servletu

Při vytvoření Java Servletu se jedná pouze o jedinou její instanci a každý uživatelský požadavek má za následek nové programové vlákno, které se spustí podle okolností doGet nebo doPost v závislosti na typu požadavku protokolu HTTP. Nyní rozebereme mnohem konkrétněji jak se Java Servlety vytvářejí a ruší, dále jak a kdy se používají různé jejich metody. (HALL, 2001)

[3] Zdroj: *Head First Servlets and JSP™ Second Edition, 2008*

O životní cyklus se stará webovský kontejner aplikačního serveru a má 4 fáze:

1. vytvoření instance třídy Java Servletu,
2. inicializace Java Servletu,
3. obsluha požadavku ze strany klientů,
4. destrukce Java Servletu.

Vytvoření instance třídy Java Servletu

Vytvoření instance třídy Java Servletu zabezpečuje webový kontejner. Standardně je pak proveden jeho konstruktor bez parametrů. Konstruktor je speciální metoda, která inicializuje objekt při jeho vytváření. Webový kontejner vytváří pouze jednu instanci třídy Java Servletu, která je určena pro paralelní obsluhu všech požadavků uživatele.(HALL, 2001)

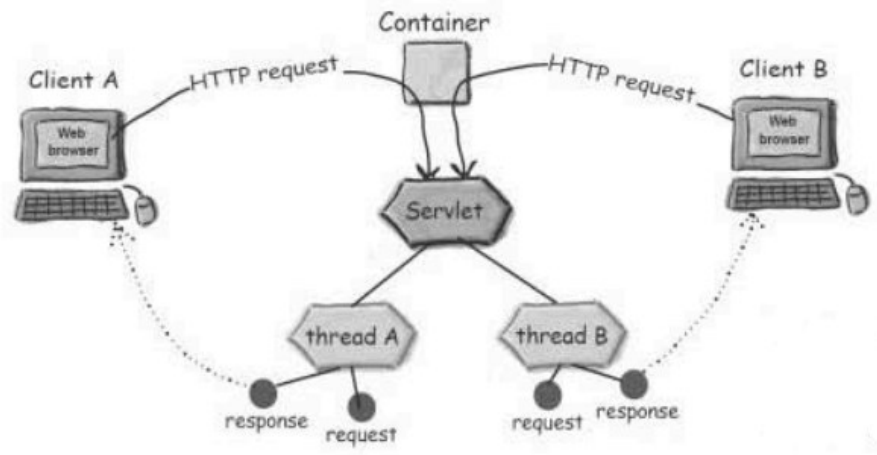
Inicializace Java Servletu

Inicializaci instance třídy Java Servletu provádí webový kontejner voláním metody init. Inicializace je prováděna, když je instance Java Servletu poprvé vytvořena, neprovádí se opakovaně pro každý uživatelský požadavek. Java Servlet je vytvářen, když je aplikační server poprvé spuštěn, a to v závislosti na tom, jak je Java Servlet registrován u webového serveru. Tato komponenta bude vytvořena pro první požadavek v případě, že není výslovně zaregistrován, ale je uložen do jednoho ze standardních adresářů serveru.

Příklady pro inicializaci jsou předběžná vyhrazení několika databázových připojení, nebo když Java Servlet musí číst specifické hodnoty nastavení serveru ještě dříve, než může dokončit inicializaci. Například musí znát hodnoty nastavení databáze, soubory hesel, nebo jiné provozní parametry serveru. (HALL, 2001)

Obsluha požadavku ze strany klienta

Jak tvrdí (HALL, 2001, str. 34) *„pokaždé když server dostane pro Java Servlet požadavek, server spustí nové programové vlákno a vyvolá metodu service. Metoda service překontroluje typ požadavku HTTP (GET, POST, PUT, DELETE) a vyvolá metodu doGet, doPost, doPut, doDelete atd. tak, jak je to potřeba.“* Jak uvádí (Basham, Sierra, & Bates, 2008) každý požadavek vytváří nové vlákno. A pro každý požadavek objekt typu požadavku (request) a odpovědi (response) jak můžeme vidět na obrázku 2-3.



Obrázek 2-3 Obsluha požadavku webovým kontejnerem [4]

Jak dále uvádí (HALL, 2001, str. 36), systém vytvoří instanci Java Servletu a pak vytvoří nové vlákno pro každý uživatelský požadavek, takže pokud přijde nový požadavek, dokud předchozí ještě neskočil, poběží zároveň několik programových vláken. To znamená, že naše metody doPost a doGet musí pozorně synchronizovat přístup k sdíleným datům, neboť několik programových vláken se může pokusit o souběžný přístup k těmto datům.

Destrukce Java Servletu

Dle (HALL, 2001) server může rozhodnout o odstranění zavedené instance Java Servletu, a to možná proto, že o to výslovně požádá administrátor serveru, nebo možná proto, že Java Servlet je po dlouho dobu nečinný. Což znamená ukončení jeho životního cyklu. Tuto fázi není možné realizovat v době obsluhy aspoň jednoho požadavku. Destrukci Java Servletu provádí pouze jedno programové vlákno. Webový kontejner volá metodu destroy a dává tím najevo, aby Java Servlet uzavřel databázová připojení, zastavil procesy na pozadí, napsal seznamy cookies (malé množství dat, která server pošle prohlížeči a ten je uloží na počítač uživatele, přičemž při každé další návštěvě téhož serveru pak prohlížeč tato data posílá zpět serveru. Cookies běžně slouží k rozlišování jednotlivých uživatelů) nebo napsal počet přístupů na disk. Poté zabezpečí destrukci instance třídy Java Servletu. (Basham, Sierra, & Bates, 2008)

Nesmíme ale zapomenout, že dokonce ani technologie Java nevyloučí někoho, kdo chce přepojovat napájecí kabel k serveru. Takže se nemůžeme spoléhat na destrukci Java Servletu jako na jediný mechanizmus pro uložení stavu na disk nebo na jiné úložiště dat.

[4] Zdroj: *Head First Servlets and JSP™ Second Edition, 2008*

„*Činnosti jako počítaní přístupů nebo akumulování seznamů hodnot cookies, které udávají speciální přístup, by měli velmi často periodicky zapisovat svůj stav na disk*.“(HALL, 2001, str. 37).

2.2.5 JSP stránky

Java Server Pages (JSP) je technologie, která umožňuje vložit Java Servlet kód přímo do textově založeného dokumentu. Podle (Jendrock, Evans, Gollapudi, Haase, & Srivathsa, 2010) JSP stránka je textový dokument, který obsahuje dva typy textu:

1. statický obsah, který je vyjádřený HTML nebo XML,
2. JSP elementy, které určují, jak má stránka zkonstruovat svůj dynamický obsah.

Jak uvádí (Basham, Sierra, & Bates, 2008) o překlad JSP stránek se stará webový kontejner. Java Servlety a JSP stránky jsou pro webový kontejner totožné.

2.2.5.1 JSTL

Java Server Pages Standard Tag Library (JSTL) zahrnuje jádro funkcí, které jsou běžné JSP aplikacím. Díky standardizaci umožňuje spuštění aplikace na jakémkoliv kontejneru, který podporuje JSTL, a je více pravděpodobné, že implementace těchto značek je optimalizovaná.

JSTL má podmínkové značky pro zajištění plynulé kontroly, značky pro manipulaci XML souborů, značky pro přístup k databázi. (Jendrock, Evans, Gollapudi, Haase, & Srivathsa, 2010)

2.2.6 JSF stránky

Java Server Faces poskytují framework pro tvorbu webových aplikací a využívají konceptu Java Servletů a JSP stránek. Více informací uvádí (Jendrock, Evans, Gollapudi, Haase, & Srivathsa, 2010, str. 26)

2.2.7 Funkce webového kontejneru

Zatím jsme zmínili, že web kontejner se stará o životní cyklus Java Servletu, kromě toho poskytuje ještě další funkce (Basham, Sierra, & Bates, 2008):

- **Komunikační podpora** – Kontejner poskytuje přímou vazbu mezi webovým serverem a Java Servletem.
- **Vícevláknová podpora** – Kontejner automaticky vytváří nové programové vlákno pro každý požadavek, který obdrží.
- **Deklarace bezpečnosti –** V rámci kontejneru lze vytvořit deployment description, což je XML soubor, který je přítomen v každé webové aplikaci a nabízí široké možnosti konfigurace, jako je např. nastavení uvítací stránky, mapování Java Servletů na různé URL adresy, filtry aj. Povinný název souboru je web. xml.
- **JSP podpora –** Kontejner se stará o přeložení JSP stránek do podoby Java Servletu.

2.2.8 Java API (Application Programming Intefrace)

Java obsahuje mnoho API určené pro různé účely. Některé jsou dostupné pouze v určitých kontejnerech. Kvůli velkému množství těchto rozhraní je v této kapitole zmíněno jen několik z nich. Především ty, které se podílejí na tvorbě systému Montážního deníku.

Pro informační systém Montážního deníku jsou použita rozhraní:

- Java Servlety
- Java Server Pages (JSP)
- Java Database Connectivity (JDBC)

2.2.9 Java DataBase Conectivity (JDBC)

JDBC poskytuje standardní knihovnu pro přístup k relačním databázím. Díky JDBC aplikačnímu programovému rozhraní můžeme získat přístup do různých relačních databází pomocí stejné syntaxe SQL. Je nutné poznamenat, že JDBC standardizuje mechanismus pro připojení k databázím, syntaxi pro odesílání dotazů a provádění transakcí a datové struktury reprezentující výsledek, ale nepokouší se standardizovat syntaxi jazyka SQL. Zkratka SQL (Structured Query Language) znamená databázový dotazovací jazyk. Rovněž lze použít

jakákoliv rozšíření SQL, která podporuje dodavatel databáze. JDBC umožňuje měnit databázová zařízení, porty, a dokonce i dodavatele databází při minimální změně kódu. Pro připojení k vybrané databázi je nutné použít specifický ovládač. (HALL, 2001)

2.2.9.1 Základní kroky při použití technologie JDBC

(HALL, 2001) uvedl ve své publikaci „*Java: servlety a stránky JSP",* že pro použití technologie JDBC existuje sedm standardních kroků:

1. zavedení ovládače,
2. definování připojovací URL,
3. navázání spojení,
4. vytvoření příkazu,
5. provedení dotazu,
6. zpracování výsledků,
7. uzavření spojení.

Zavedení ovladače

Ovládač je software, který umí komunikovat s databázovým serverem. Zavedení ovladače je zavedení instance příslušené třídy ovladače. Jak tvrdí (HALL, 2001, str. 474), „*statický inicializátor v samotné třídě vytvoří automaticky instanci ovladače a registruje ji pomoci správce ovladačů JDBC*." Metoda Class.forName zabezpečí zavedení třídy, aniž bychom vytvořili její instanci. Tato metoda používá řetězec obsahující plně kvalifikovaný název třídy (název, který zahrnuje názvy balíků) a zavádí odpovídající třídu. Musíme poznamenat, že volání této metody může vyvolat výjimku ClassNotFoundException, takže by se měla nacházet v bloku try/catch.

Pro připojení k relační databází MySql o které bude dále řeč, se používá *Connector/J*, který je takzvaný JDBC ovladač čtvrtého typu, což znamená, že je celý napsán v Javě a očekává přítomnost JDBC. Na stránce dev.mysql.com je k dispozici stabilní verze *Connector/J 5.0*, která podporuje funkce MySQL 5.0. (KOFLER, 2007)

Definování připojovací URL

Po zavedení ovladače JDBC musíme specifikovat umístění databázového serveru prostřednictvím tzv. URL řetězce. URL vztahující se na databáze obsahuje serverový hostitel, port a název databáze.

Navázání spojení

Pro skutečné navázání síťového spojení předáváme URL, uživatelské jméno a heslo pro danou relační databázi metodě getConnection třídy DriverManager. Musíme poznamenat, že provádění metody getConnection může vyvolat výjimku SQLException, takže musíme použít blok try/catch.

Možnou součástí tohoto kroku je vyhledání informací o databázi pomoci metody getMetaData třídy Connection. Další užitečné metody ve třídě Connection zahrnují prepareStatement (vytvoří objekt implementující rozhraní PraperedStatement), prepareCall (vytvoří objekt implementující rozhraní CallableStatement), rollback, commit, close (končí spojení) a isClosed (ptá se, jestli vypršel čas spojení nebo byl explicitně uzavřen). (HALL, 2001)

Vytvoření příkazu

Objekt Statement se používá pro odesílání dotazu a příkazů databázi a je vytvořen na základě třídy Connection, tato instance reprezentuje konkrétní navázanou konektivitu s vybranou databází prostřednictím JDBC ovladače.

Provedení dotazu

Pokud je k dispozici objekt Statement, můžeme ho použít pro odeslání dotazů SQL pomocí metody executeQuery, která vrátí objekt typu ResultSet, jejichž výsledkem je vybraná tabulka příkazů SQL jazyka typu SELECT.

Pro modifikaci vybrané tabulky databáze používáme exexuteUpdate, která používá dotazy typu UPDATE, INSERT nebo DELETE a vrací výsledný počet řádků tabulky, které jsou vloženy, zrušeny, příp. změněny prováděním příslušného SQL příkazu. (HALL, 2001)

Zpracování výsledků

Nejjednodušší způsob jak pracovat s výsledky je jejich zpracování po řádku, pomocí metody next třídy ResultSet, která pro procházení jednosměrné sady v tabulce a vrací hodnotu typu boolean. Pokud je proveden posun za poslední řádek tabulky, je vrácena hodnota false.

Uzavření spojení

Pro uzavření spojení explicitním způsobem se používá metoda close. Tento krok se odkládá, pokud očekáváme další databázové operace, neboť režie otevření spojení je obvykle

velká. Ve skutečnosti je opětovné použití existujícího spojení tak důležitou optimalizací, že je vytvořena knihovna právě pro tento účel.

2.2.10 Sdílení spojení (connection pooling)

Otevření spojení k databázi je procesem náročným na čas. Pro krátké dotazy může otevření spojení trvat mnohem déle než skutečné vyhledávání v databázi. Proto má smysl opětovně použít objekty Connection v aplikacích, které často přistupují ke stejné databázi. Z této třídy značně těží Java Servlety a stránky JSP, neboť databáze, ke které se připojuje nějaký Java Servlet nebo stránka JSP, je typicky předem známa (např. je specifikována v metodě init). Postup pro vytvoření sdílení spojení nalezneme v příloze 3.

2.3 Informační systém

Cílem práce je vytvoření informačního systému Montážního deníku. Každý informační systém pracuje s informacemi, a tedy s daty. Je logické, že data musí být někde uložena. V našem případě jsou data uložena v relační databázi MySql.

Podle (KOFLER, 2007, str. 34) „*databáze je uspořádaná množina dat, která je normálně uložena v jednom nebo několika datových souborech. Data jsou strukturovaná v tabulkách, s možnými referencemi mezi tabulkami. Existence těchto relací mezi tabulkami vedla k pojmu relační databáze*.“

Příkladem relačních databázových systému jsou MySql, Oracle, Microsoft SQL Server, IBM DB2. Takový systém obsahuje programy pro řízení relačních databází. Mezi úkoly pro relační databázové systémy nepatří jen bezpečné skladování dat, ale i úkoly jako vykonávání příkazů pro vyhledávání, analyzování a třídění existujících dat, a také ukládání nových dat. Místo o databázovém systému často mluvíme o databázovém serveru. (KOFLER, 2007)

Pokud máme servery, máme také i klienty. Každý program připojený na databázový systém nazýváme databázový klient. Databázoví klienti mají za úkol zjednodušit koncovým uživatelům práci s databází. Uživatel právem očekává pohodlnou práci pro nalezení a vkládání dat, rozbalovací výběr a další pomůcky pro usnadnění práce. Databázoví klienti se vyskytují v mnoho formách a často ani uživatel netuší, že pracuje s databázovým programem. V našem případě jsou to Java Servlety a JSP stránky.

2.3.1 Typy databázových systémů

Mezi základní databázové systémy patří:

- relační databáze,
- objektově-orientované databáze, taková databáze může ukládat samostatné objekty. Ačkoli byl v posledních letech trend na rozšiřování objektově-orientovaných programovacích jazyků, objektově-orientované databáze nalezly na trhu jen malé místo. (KOFLER, 2007)

Musíme poznamenat, že k relačním databázím můžeme přistupovat prostřednictvím objektově-orientovaných programovacích jazyků. To však neznamená, že tím změníme relační databázi na objektově-orientovanou.

2.3.2 Návrh informačního systému

Jak tvrdí (Kaluža & Kalužová, 2012) data (jejich struktura i obsah) a příslušné algoritmy tvoří podstatu informačního systému. Modelování dat a jejich algoritmů znamená modelování informačních systémů. Proces pro vytvoření datového modelu se historicky vyvíjel od funkčního přístupu, přes datový, až objektovému přístupu. V práci jsme využili datový přístup, který se soustředí na prvotní vytvoření celkového datového modelu a pak se zabývá jednotlivými funkcemi.

2.3.3 Datové modelování

Pro zajištění integrity a odstranění redundance dat v databázi je třeba provést datové modelování. Výsledkem je databázové schéma, které slouží k popisu struktury dat. K tomuto účelu se používá hlavně takzvaná tříúrovňová koncepce datového modelování, která zahrnuje sémantickou, konceptuální a logickou úroveň procesu.

2.3.4 Sémantický datový model

V úrovni sémantického modelování je cílem nalezení typů objektů odrážejících modelovanou realitu. Zkoumání objektivní reality přináší identifikaci určitých předmětů hmotné i nehmotné povahy, které jsou významné z hlediska vyvíjeného systému. Metodou, kterou se tyto prvky transformují do typů objektů, je metoda abstrakce. Podle (Kaluža & Kalužová, 2012) existují tři typy abstrakce:

- **Klasifikace** – je abstrakce, která je základním konstruktorem odrážející objektivní realitu a aplikuje se pro identifikaci typů objektů.
- **Agregace** – definuje nový typ objektu z množiny typů objektů, které se stanou jeho komponentami. Např. z komponent jméno, věk, adresa. Lze vytvořit agregací typ objektu zaměstnanec.
- **Generalizace** – je třetím typem abstrakce, která definuje vztah podmnožiny mezi výskyty dvou nebo více typů objektů.

Abstrakcí do soustavy typů objektů se provádí v zásadě ve třech krocích:

1. Identifikace vstupních datových požadavků.
2. Specifikace typů objektů a jejich charakteristik.
3. Revize struktury typů objektů.

2.3.4.1 Identifikace vstupních datových požadavků

Prvotním úkolem v procesu datového modelování je vymezení cíle a rozsahu řešení datového modelu. Na vymezení cíle a rozsahu řešení bezprostředně navazuje identifikace všech datových požadavků příslušejících k řešenému systému. K tomu se využívá metody rozhovoru nebo jejich kombinací, dotazníkového šetření, rozboru písemných materiálů, pozorování. Shromáždí se tak dokumentace zahrnující:

- slovní popisy,
- platné vstupní i výstupní formuláře,
- formáty stávajících datových struktur.

Důležité je aby, materiály byly co nejkomplexnější.

2.3.4.2 Specifikace typů objektů a jejich charakteristik.

Věcnou analýzou materiálů se specifikují jednotlivé typy objektů formující datovou strukturu. V této části procesu je specifikace typů objektů a jejich charakteristik, kdy se určí typ objektu (např. objednávka) a jeho charakteristiky (číslo objednávky, datum vystavení, položky objednávky).

2.3.4.3 Revize struktury typů objektů.

Posledním krokem je revize struktury datového modelu. To znamená, že se identifikují negativní rysy modelu, jimiž jsou především:

- synonyma a homonyma typů objektů a jejich charakteristik,
- redundance typů objektů a jejich částí,
- rozporné definice stejných prvků objektivní reality.

Typy objektů a jejich charakteristik ve struktuře následně opraví tak, aby se minimalizovaly zjištěné nedostatky. Výsledná struktura typů objektů je východiskem pro konceptuální modelování. Cílem sémantického modelování je co nejúplnější strukturovaný popis datové části řešeného projektu informačního systému.

2.3.5 Konceptuální datový model

Konceptuální datový model je založen na grafickém vyjádření. Metoda pro konceptuální modelování je metoda E-R diagramu prezentovaná poprvé v roce 1976 Petrem Chenem. V roce 1988 byla převzata metoda E-R americkým standardizačním institutem ANSI jako standard. Postupem docházelo ke zdokonalování této metody tak, aby lépe postihovala modelované situace. Dnes je metoda E-R jednou ze dvou nejpoužívanějších ke konceptuálnímu modelování.

Druhou metodou je diagram tříd tvořící součást metodiky UML. Autory UML jsou P. Boochem, J. Rumbaughem a I. Jacobsonem vytvořena v roce 1995. (Kaluža & Kalužová, 2012)

Pro tvorbu E-R diagramu jsou základní konstrukty:

- entita,
- vztah,
- atribut,
- doména,
- klíč.

Dále přikročíme ke specifikaci jednotlivých konstruktů E-R diagramu.

2.3.5.1 Entita

Reprezentuje typ objektu reálného světa, který se liší od ostatních objektů a je schopen nezávisle na ostatních existovat. Jméno entity by mělo být vyjádřeno výstižně podstatným jménem. Je nutné jednoznačně oddělit typ od jednotlivých objektů, tedy v prezentovaném

pojetí entitu od výskytu entity (např. u entity Zaměstnanec může jít o výskyt ‚Novák', u entity Výrobek o výskyt ‚dlaždice'). Výskyt entity je tedy množina stejných objektů.

Entita může být silná nebo slabá podle toho, jak závisí na existenci jiné entity. Pokud je entita silná, tak svým primárním klíčem nezávisí na žádné entitě, u slabé entity neexistuje žádný atribut, který by ji identifikoval. Graficky se slabá entita v datovém modelu odlišuje od silné entity neuvedením žádného primárního klíče, protože v datovém modelu může být slabá entita propojena s dalšími entitami, vyznačí se vazba na silnou entitu zdvojenou čarou. (Kaluža & Kalužová, 2012).

2.3.5.2 Vztah

Vztah je vazba mezi dvěma, nebo více entitami. Nejběžnějším typem vztahu, je tzv. **asociativní vztah**. Tento vztah reprezentuje asociace jedné nebo několika entit. Každý asociativní vztah je charakterizován třemi základními charakteristikami: stupněm, kardinalitou a volitelností.

Stupeň

Stupněm vztahu se rozumí počet entit asociovaných v jednom vztahu. Nejnižší je stupeň jedna, když vztah se váže k jedné entitě, jde o **unární** nebo také rekurzívní vztah. Vztah druhého stupně mezi dvěma entitami je **binární**, mezi třemi **ternární**. Podobně lze specifikovat vztah čtvrtého až n-tého stupně tzv. **n-ární** vztah, avšak v datových modelech je jejich praktický výskyt mizivý.

Kardinalita

Kardinalita vztahu vyjadřuje počet výskytů entit účastnících se jednoho výskytu vztahu. Mezi dvěma tabulkami existují tři typy vztahů:

- **1:1 -** Ve vztahu „jeden na jednoho" znamená, že ke každému výskytu entity z první entity existuje výskyt entity v druhé entitě a naopak (např. jedna stavba souvisí s jednou zakázkou v entitě zakázka). To znamená, že dvě entity používají stejný primární klíč. Tyto vztahy jsou velmi vzácné, protože informace v takových entitách bychom mohli vměstnat do jedné entity. Značnou nevýhodou oddělených entit je, že se musíme starat o to, aby entity zůstaly synchronní.
- **1:N –** Ve vztahu „jeden na mnoho" znamená, že každého výskytu entity v první entitě může existovat několik výskytů entity v druhé entitě (např.

prodejce může mít několik odběratelů). Tento vztah je velmi užitečný, protože eliminuje duplicitní data.

- **N:M –** Zde ke každému výskytu první entitě může existovat několik výskytů entity z druhé entity a naopak. (Např. v jedné objednávce může být několik typů zboží a daný typ zboží může být na několika objednávkách.)

Volitelnost

Je třetí charakteristikou vztahu a vyjadřuje, zda vztah je povinný či volitelný ze strany jedné či druhé entity, tedy zda každému výskytu vztahu musí nebo může odpovídat jeden nebo několik záznamů příslušné entity.

2.3.5.3 Atribut

Reprezentuje vlastnosti tabulky nebo vztahu. Každý atribut nabývá určitých konkrétních hodnot. Například v entitě *Student* jsou atributy *jméno, příjmení, datum narození.* Atributy mohou být:

- Jednoduché atributy – Skládají se z jedné komponenty, tedy nabývají hodnot, které jsou dále nerozložitelné.
- Složené atributy – Tvořené více komponentami, které mají společný význam. (Například atribut *adresa* je složena z jednoduchých komponent *psč, město, ulice.)*

2.3.5.4 *Doména*

Dle (Kaluža & Kalužová, 2012, str. 47) se doménou rozumí „*množina přípustných hodnot přiřazená jednomu, nebo více atributům. Například množina všech hodnot příjmení (doména) může být přiřazena k atributu příjmení v entitě Zaměstnanec, ale též ke stejnojmennému atributu v entitě Pojištěnec.*“

2.3.5.5 Klíč

Klíč je jedním nebo několika atributy identifikujícími výskyty dané entity. Slouží-li jediný atribut, nazývá se jednoduchým klíčem, je-li využito více atributů, hovoříme o složeném klíči. Vztahy úzce souvisí s primárními a cizími klíči.

Primární klíč

Úkolem primárních klíčů je urychlit vyhledání příslušného záznamu v tabulce. Takovou operaci potřebujeme, když sbíráme data z několika tabulek, což je poměrně často.

Ve většině databázových systémů je možnost složit primární klíče z několika atributů tabulky. Ať se primární klíč skládá z jednoho, nebo několika atributů platí pro něj následující pravidla:

- Primární klíč musí být unikátní. Není možné, aby dva různé výskyty entity měly stejné hodnoty v atributu primárního klíče.
- Primární klíče musí být kompaktní, existují pro to dva důvody:
    - Pro zajištění maximalizované rychlosti vyhledávání.
    - Hodnoty primárního klíče mohou sloužit jako hodnoty cizího klíče v jiných entitách, a ty musí být kvůli efektivitě také kompaktní.

Cizí klíč

Takto se označuje atribut, který je v jiné tabulce použit jako primární klíč. Úkolem cizích klíčů je odkazovat se na záznamy hlavních tabulek.

2.3.6 Logický relační model

Právě jednoduchost datového modelu i práce s daty je rysem relační koncepce, který způsobil její dnešní dominanci. Nástupem relační koncepce usnadnil technologický zlom vyvolaným nástupem osobních počítačů. Dle (Kaluža & Kalužová, 2012) relační datové modelování přebírá některé konstrukty, které jsou definovány v konceptuální úrovni, mezi které patří atribut, doména, klíč a zavádí zcela nový pojem relace.

2.3.6.1 Definice relace

Relací je podle (Kaluža & Kalužová, 2012, str. 73) „*dvourozměrná datová strukturu tvořena záhlavím a tělem relace. Záhlaví relace je pak definováno jako množina dvojic (A_i, D_i), kde atribut A_i je přiřazen právě jedné doméně D_i, pro i = 1,2,…,n; všechna A_i musí byt vzájemně odlišná. Tělo relace je tvořeno množinou n-tic, které jsou množinami dvojic (A_i, v_{ji}), kde A_i je i-tý atribut a dále v_{ji} je j-tá hodnota z domény D_i pro j=1,2,…,m, kde m je počet n-tic v množině; m je pak kardinalitou a n stupněm relace (pro n=1 se hovoří o unární relaci, pro n=2 o binární, atd.).*“

Existuje řada odlišných pojetí v terminologii i v definici uvedených konstruktorů. Jak uvádí (KOFLER, 2007) relaci můžeme chápat jako tabulku, n-tic používá jako datový záznam, nebo zjednodušeně záznam. Dále používá termín sloupec namísto atributu.

Mezi základní vlastnosti relace patří (Kaluža & Kalužová, 2012):

- neexistence duplikátu n-tic,
- libovolné pořadí atributů,
- libovolné pořadí n-tic,
- nerozložitelnost hodnot atributů.

První tři vlastnosti poukazují na fakt, že tělo relace je množinou v matematickém smyslu, v posledním případě se hovoří o atomičnosti atributů, což znamená, že hodnoty žádného atributu nelze rozložit, aniž by došlo ke ztrátě informace. Terminologicky se relace, která neobsahuje vícehodnotové atributy, označuje v **první normální formě**.

2.3.6.2 Normalizace

Pro odstranění anomálií v datovém modelu je proces zvaný normalizace dat. Všechny relace tvořící datový model se podrobí testům na normalizaci. Normalizační procedury prověří strukturální správnost a konzistenci vnitřních částí modelu, nikoliv však správnost celkové struktury a přesnost odrazu model modelované reality. Normalizace také neřeší optimalizaci datové struktury z hlediska fungování databáze. Na databázi se postupně aplikují tří normální formy.

První normální forma

Pravidla pro první normální formu:

- Musíme eliminovat atributy se stejným obsahem.
- Relaci musíme vytvořit pro každou skupinu příslušných dat.
- Každá n-tice musí být jednoznačně identifikována primárním klíčem.

Druhá normální forma

Pravidla druhé normální formy:

- Kdykoliv se opakují obsahy některých sloupců v různých n-ticích, měla by relace být rozdělena na několik relací.
- Tyto relace musíme propojit s cizími klíči.

Třetí normální forma

Třetí normální forma má jen jedno pravidlo:

- Atributy, které nejsou přímo závislé na primárním klíči, musíme odstranit (přesunout je do samostatné relace).

2.3.7 Návrh metodiky relačního modelování

Po vytvoření konceptuálního modelu, který je prezentován ve tvaru E-R diagramu, následuje transformace tohoto modelu do logické relační normy. Následující postup vyjadřuje logiku prvotního vytvoření neúplných definic relací, což jsou tzv. předběžné relace specifikované jménem, klíčem a případně cizími klíči, jejich kompletace zbývajícími atributy, normalizace a určení doménových charakteristik.

Tento relační model je spjatý s konkrétním databázovým systémem, který je v našem případě MySQL.

2.4 MySQL

MySQL je relační databázový systém s architekturou klient/server. Je dostatečně stabilní a bezpečný pro mnoho aplikací a nabízí výborný poměr cena/výkon nejenom protože MySQL je zadarmo, ale také proto, že má mírné nároky na hardware. MySQL je považován téměř za standard v oblasti internetových aplikací. Vlastnosti MySQL jsou veřejně dostupné na stránce http://dev.mysql.com/doc/refman/5.5/en/. (KOFLER, 2007)

3 Analýza a popis stávajícího systému

Firma Elektro Heinich sídlící ve Vratimově vznikla v roce 1992, působí především v Moravskoslezském kraji a zabývá se veškerou elektroinstalací, montáží antén a satelitů, elektro-revizemi, montáží topných systémů, osvětlením včetně návrhů atd.

V současnosti jsou všechny informace o činnostech zakázek zapsané v papírové podobě v Montážním deníku v podobě sešitu. Tento Montážní deník zachycuje dílčí práce technicky jednoduché, záznamy o postupu prací a jejich souvislostech od doby zahájení prací na staveništi do doby jejího dokončení, popř. do odstranění vad a nedodělků zjištěných při kontrolní prohlídce stavby, v podobě jednoduchého záznamu. Tento jednoduchý záznam o stavbě v Montážním deníku musí být veden v časových intervalech, které zachycují reálný průběh prací či výstavby.

Jednoduchý záznam o stavbě obsahuje:

1) záznamy o průběhu provádění stavebních a stavebně montážních prací a o skutečnostech ovlivňujících zhotovení díla,
2) záznamy o mimořádných událostech během výstavby.

Dále se zaznamenávají činnosti a okolnosti, které mají vliv na:

3) postup prací a použití materiálů,
4) bezpečnou instalaci a užívání technického vybavení a funkčních dílů stavby,
5) revize elektrozařízení, zkoušky a revize plynových zařízení, kouřovodů, komínů apod.,
6) podmínky bezpečného provádění stavby a ochrany zdraví při práci,
7) dodržení údajů obsažených v ohlášení stavby včetně ověřené projektové dokumentace, případně nutnost drobných odchylek od ní.

Cílem práce je nahrazení papírového montážního deníku elektronickým ve formě informačního systému. Ten bude plnit funkci elektronického montážního deníku a bude zabezpečovat okamžitou a přehlednou dostupnost veškerých podkladů pro řízení montáže, a to kdekoliv a kdykoliv všem oprávněným osobám. Velkou výhodou elektronického montážního deníku je i odstraňování, přepisování dokumentů, a tím zajištění nežádoucích chyb – překlepy, přehlednutí, apod. Tento infomační systém bude sloužit pouze firmě Elektro Heinich.

Rozhovorem s výkonným pracovníkem firmy, byly zaznamenány podmínky, které má daný informační systém splňovat:

1. Infomační systém bude ve formě webové aplikace s přístupem pro více uživatelů.
2. Každý uživatel jednak uvidí informace o sobě a jednak bude zapisovat hodiny k danému záznamu se souhlasem výkonného pracovníka.
3. Výkonný zaměstnanec dané zakázky může pracovat se všemi informacemi v dané zakázce a přidávat jednotlivé zaměstnance k danému záznamu.
4. Administrátor aplikace bude mít přístup ke všem informacím s možností vytvářet nové uživatele.

4 Návrh koncepce řešení pro vývoj webové aplikace

Pro vývoj webové aplikace je prvním krokem návrh databáze. Tento krok má velký vliv na efektivitu programu, obtížnost programování, údržbu a flexibilitu. Chyba v návrhu má katastrofální důsledky na možnost opravování chyb v již hotovém programu. Pro vytvoření databáze se vychází z datového modelování. Aby bylo datové modelování co nejpřesnější, je vhodné použít tříúrovňovou koncepci zpracování.

4.1 Tříúrovňová koncepce

Informace z reality mohou být získány různými způsoby. Mezi nejzákladnější patří zkoumání formulářů, dotazníků a jiných textových materiálů, nebo rozhovorem s respondentem za účelem zjistit, které objekty jsou pro vývoj aplikace důležité. V tomto případě jako zdroj informací posloužil Montážní deník výše zmiňované firmy a informace od respondenta, ze kterých vznikly následující objekty.

Na základě analýzy obsahu uvedeného formuláře v příloze 1 lze definovat následující typy objektu v sémantickém datovém modelu.

4.1.1 Sémantický datový model

V této úrovni jsou objekty a jejich charakteristiky. Jedná se o abstrakci reality pro objekty, které vyjadřují modelovanou realitu. Tyto objekty budou později převedeny na konceptuální model, přičemž se neberou v potaz primární klíče a vztahy mezi objekty a modelují se pouze prvky, které jsou pro pozdější vývoj podstatné.

Typ objektu: Stavby

Popis: Identifikační údaje stavby, konkrétně firmy, které budou využívat služeb elektro Heinich.

Charakteristiky: název stavby, plánovaná realizace datum zahájení a ukončení, vlastník vytvořené stavby.

Typ objektu: Zakázky

Popis: Identifikační údaje o zakázce, která se týká jednotlivé stavby.

Charakteristiky: předmět díla, smlouva, objednavatel, oprávněný pracovník, výkonný pracovník, plánovaná realizace zahájení a ukončení, bezpečnost práce.

Typ objektu: Záznamy

Popis: Identifikační údaje o záznamech, které jsou obsaženy pod jednou zakázkou.

Charakteristiky: Datum, záznam.

Typ objektu: Zhotovitel

Popis: Údaje o zhotoviteli, který zhotovuje právě jeden záznam, zhotovitelů může být pod jedním záznamem více. (Např. v daném záznamu pracuje více zaměstnanců a zaměstnanec se při realizaci daného záznamu označuje jako zhotovitel.)

Charakteristika: čas zahájení, čas ukončení, počet hodin.

Typ Objektu: Zaměstnanec

Popis: Údaje o zaměstnanci, který se podílí na zhotovení dané práce právě jako jeden ze zhotovitelů.

Charakteristika: jméno, příjmení, telefon.

Typ Objektu: Uživatelé

Popis: Údaje o uživateli, který využívá daný informační systém montážního deníků, pouze zaměstnanec může být uživatelem. Vyznačená role určuje v informačním systému oprávnění, které zaměstnanec obdrží.

Charakteristika: přihlašovací jméno, přihlašovací heslo, role.

4.1.2 Konceptuální datový model

Tato kapitola navazuje na rozvinutí procesu sémantického modelování do pokročilejší konceptuální úrovně. Principem je převést všechny typy objektů do entit a znázornit vztahy mezi sebou. Způsob znázornění konceptuálního datového modelu je graficky v E-R digramu, kde jsou základními komponentami entita a vztah.

4.1.2.1 Specifikace konceptuálního modelu

Entita: Stavby **Identifikační označení**: Stavby

Atribut	**Identifikační označení**
Číslo zakázky	#cZakazky
Název stavby	nazevStavby
Vlastník vytvořeného deníku	vytvoril

Entita: Zakázka **Identifikační označení:** Zakazka

Atribut	**Identifikační označení**
Číslo zakázky	#cZakazky
Předmět díla	predmetDila
Smlouva	smlouva
Objednavatel	objednavatel
Oprávněný pracovník	opravnenyPrac
Zahájení	zahajeni
Ukončení	ukonceni
Bezpečnost práce	bezpecnostPrace

Entita: Záznamy **Identifikační označení**: Zaznamy

Atribut	**Identifikační označení**
Datum	datum
Záznam	zaznam
Identifikační číslo zhotovitele	#idZhotovitele

Entita: Zhotovitel **Identifikační označení**: Zhotovitel

Atribut	**Identifikační označení**
Datum	datum
Čas zahájení práce	casOd
Čas ukončení práce	casDo
Počet hodin	hodiny

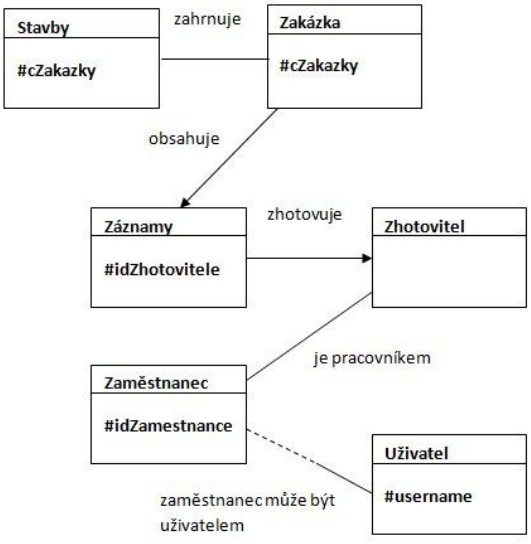
Entita: Zaměstnanec **Identifikační označení**: Zamestnanec

Atribut	**Identifikační označení**
Identifikační číslo	#idZamestnance
Jméno	Jmeno
Příjmení	Prijmeni
Telefon	telefon

Entita: Uživatele **Identifikační označení**: Uživatele

Atribut	**Identifikační označení**
Přihlašovací jméno	#Username
Přihlašovací heslo	passwrd
Role	Role (admin, ceo, member)

4.1.2.2 E-R Diagram



Obrázek 4-1 E-R diagram

4.1.3 Vytvoření relačního modelu

V této části se převádí konceptuální model do relačního. Právě zde je potřeba upravit model tak, aby bylo možné jej transformovat do definičního jazyka databáze. Entita je převedena do relace, kterou chápeme jako tabulku s odpovídajícími atributy, které můžeme označit jako sloupce. V tabulkách se určí primární klíč, poté cizí klíč, který určuje vztahy mezi jednotlivými tabulkami. U vztahu 1: N se primární klíč přidá z jedné tabulky do druhé, kde se označuje jako cizí klíč. Vztahy N: M je nutné pomocí pomocné tabulky rozbít na dva vtahy 1: N. Vztah 1:1 je třeba určit, zda je vztah povinný nebo volitelný.

Předběžné relace

Stavby(cZakazky#,…)

Zakázka(cZakazky#,cZakazky(c.k),…)

Záznamy(idZhotovitele#,cZakazky(c.k),…)

Zhotovitel(idZhotovitele(c.k),idZamestnance(c.k)…)

Zaměstnanec(idZamestnance#,username(c.k),…)

Uživatele(username#,idZamestnance(c.k),…)

Úplné relace

Stavby(cZakazky#,nazevStavby,datumOd,datumDo,vytvoril,vytvoreno)

Zakázka(cZakazky#(c.k),PredmetDila,smlouva,vykonnyPrac,bezpecnostPrace,objednavatel,opravnenyPra,zahajeni,ukonceni)

Záznamy(idZhotovitele#,cZakazky(c.k),datum,zaznam,vlozil)

Zhotovitel(idZhotovitele(c.k),idZamestnance(c.k),cas_od,cas_do,prestavka,hodiny)

Zaměstnanec(idZamestnance#,username(c.k),jmeno,prijmeni,telefon)

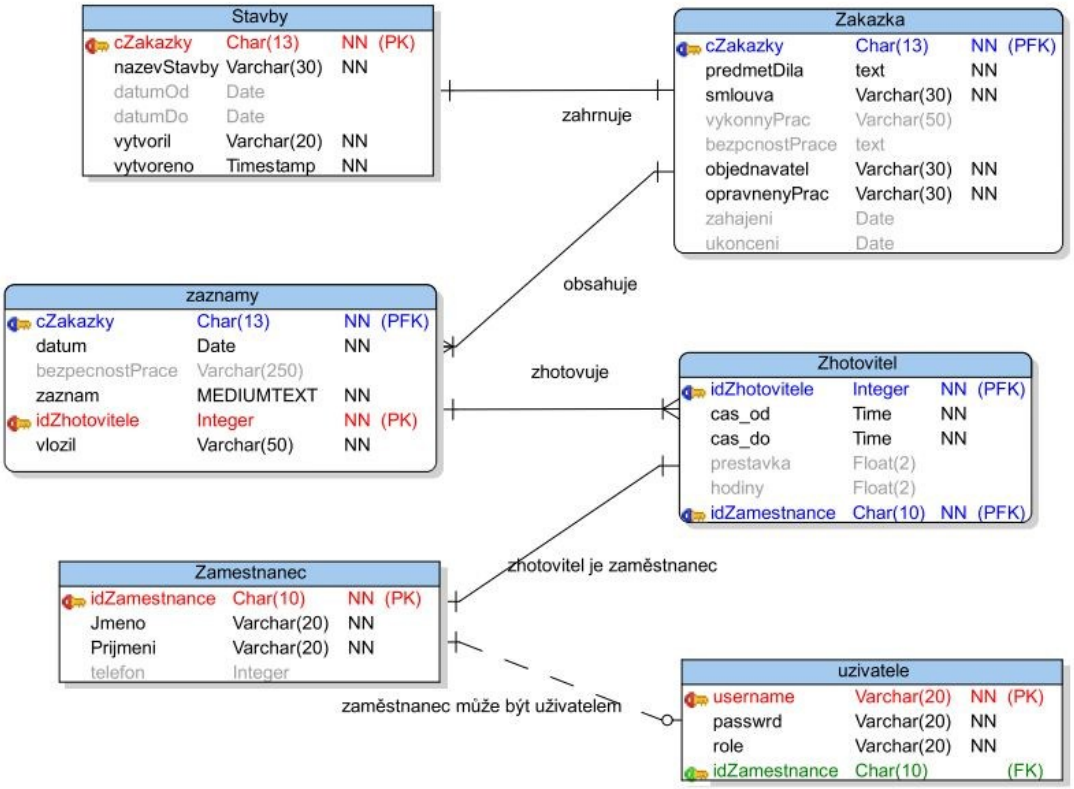
Uživatele(username#,idZamestnance(c.k),passwrd,role)

Specifikace doménových charakteristik

Název	Atribut	Datový typ	Délka	Klíč	Přípustnost null hodnot
Stavby	cZakazky	Char	13	PK	ne
	nazevStavby	Varchar	30		ne
	datumOd	Date			ano
	datumDo	Date			Ano
	Vytvoril	Varchar	20		ne
	Vytvoreno	Timestamp			ne
Zakazka	cZakazky	char	13	CK,PK	ne
	PredmetDila	Text			ne
	Smlouva	Varchar	50		Ne
	vykonnyPrac	Varchar	50		Ano
	BezpecnostPrace	Text			Ano
	Objednavatel	Varchar	30		Ano
	opravnenyPrac	Varchar	20		Ne
Zaznamy	cZakazky	Char	13	CK	ne
	Datum	Date			ne
	Zaznam	mediumText			Ne
	idZhotovitele	Int		PK	ne
	Vlozil	Varchar	50		ne
Zhotovitel	idZhotovitele	Int		CK	ne
	Cas_od	Time			ne
	Cas_do	Time			ne
	Prestavka	Float			Ano
	Hodiny	Float			Ano
	idZamestnance	Char	10	CK	Ne
Zamestnanec	idZamestnance	Char	10	PK	Ne
	Jmeno	Varchar	20		Ne
	Prijmeni	Varchar	20		Ne
	Telefon	Int			Ano
	Username	Varchar	20		Ano
Uzivatele	Username	Varchar	20		Ne
	Passwrd	Varchar	50		Ne
	Role	Varchar	20		Ne
	idZamestnance	Char	10		ano

Tabulka 1 Specifikace domén

Na obrázku 4-2 vidíme logickou propojenost modelu společně s cizími klíči, proto lze zde pozorovat, jak jsou tabulky propojeny. V každé tabulce je vedle názvu atributu i její datový typ. Tento model byl vytvořen v programu Toad Data Modeler Freeware 4.2.



Obrázek 4-2 Grafické znázornění logického modelu

4.2 Realizace v MySQL

Veškerá realizace databáze je provedena v programu HeidiSQL, přičemž tato Windows aplikace slouží pro správu MySQL. To nám umožňuje prohlížet a upravovat data, dále vytvářet a spravovat tabulky, pohledy, procedury, triggery a naplánované akce. Také můžeme exportovat strukturu a data do souboru SQL. Prvním krokem pro vytvoření naší databáze je navázat spojení s databázovým serverem. Úplný postup vytvoření databáze nalezneme v příloze 2. Jelikož žádný uživatel databázového systému by si určitě nepřál komunikovat přímo s databázovým serverem. Bylo by to příliš abstraktní a nepohodlné, kdyby uživatel musel psát kódy jazyka SQL pro komunikaci s databázovým systémem. Jelikož uživatel právem očekává pohodlné tabulky, rozbalovací výběry a další pomůcky pro nalezení a vkládání dat. Z tohoto důvodu pro zabezpečení pohodlnější práce s databází je v našem případě vytvoření webové rozhraní.

4.2.1 Návrh webového rozhraní

Webové rozhraní je nejjednodušší prostředek sdělování informací, protože webový prohlížeč má prakticky každý uživatel připojený k Internetu. Z tohoto důvodu musí každá platforma pro budování informačního systému, kterým je rovněž systém Montážního deníku, poskytovat možnost vytvoření webového klienta.

Platforma Java EE 6, která byla zmíněna v kapitole 2.2, poskytuje možnost realizace aplikační logiky, která slouží pro manipulaci s daty pomocí Java Servletu a prezentační vrstvu pro zobrazení dat pomocí JSP stránek.

V další části práce jsou popsány základní komponenty webové aplikace, jejich funkce.

4.2.2 Vytvoření webové aplikace

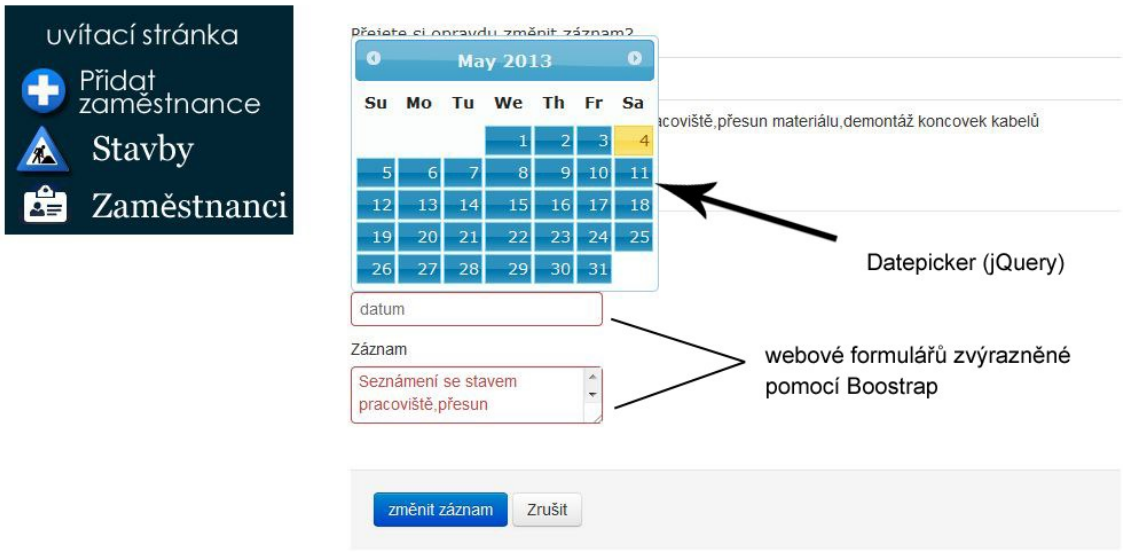
Cílem webového rozhraní je přehledně zobrazit data Montážního deníku. Webové rozhraní generuje HTML stránky a posílá je klientům, což jsou webové prohlížeče. Stránky jsou generovány dynamicky pomocí technologie JSP. Funkce pro práci s databází umožňují Java Servlety. Pro elegantní vzhled webové aplikace byla použita notace CSS (Cascading Style Sheets), která vznikla roku 1997 a je kolekce metod pro grafickou úpravu webových stránek.

4.2.2.1 Grafický návrh

Vzhled webové aplikace je tvořen obrázky a CSS, které umožňuje definovat vlastnosti určitých elementů např. barvy textu, pozadí, velikost písma, ale také i určuje polohu zobrazení objektů na stránce (obrázek, tabulka, text). Poloha zobrazení je nastavená na hlavičku stránky (logo stránky), dále na její obsah a nakonec na patičku stránky, které znázorňuje ukončení webového obsahu viz obrázek 4-3.

Obrázek 4-3 Grafické zobrazení webového rozhraní

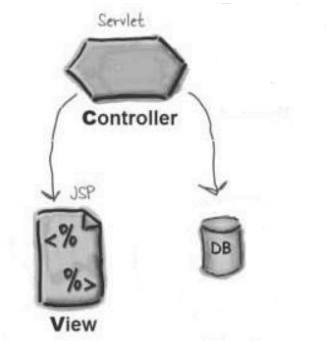
Pro zajištění elegantního vzhledu byly použity CSS Bootstrap, které jsou volně stažitelné na stránkách http://twitter.github.io/bootstrap/index.html. Tyto CSS umožňují zvýraznění webových formulářů, které jsou nezbytným doplňkem každé webové aplikace, protože bez webových formulářů nelze přihlásit do infomačního systému nebo vložit informace do databázového sytému. Také dovolují elegantní zobrazení tabulek, menu stránek a tlačítek. Další doplňující technologie je jQuery, což je javascriptová knihovna s širokou podporou prohlížečů, která klade důraz na interakci mezi JavaScriptem (Skriptovací jazyk, který umožňuje vytvořit hodiny, hodnotit data ve formuláři, počítat, dynamizovat data, umožňuje tvorbu všemožných prvků k oživení webu, přes blikající texty po jednoduché hry.) a HTML. Z technologie jQuery je použitá komponenta Datepicker, která zabezpečuje rozbalovací okno pro vložení datumu. Také volně stažitelná na http://jqueryui.com. Obě technologie můžeme vidět na obrázku 4-4.



Obrázek 4-4 Pomocné technologie

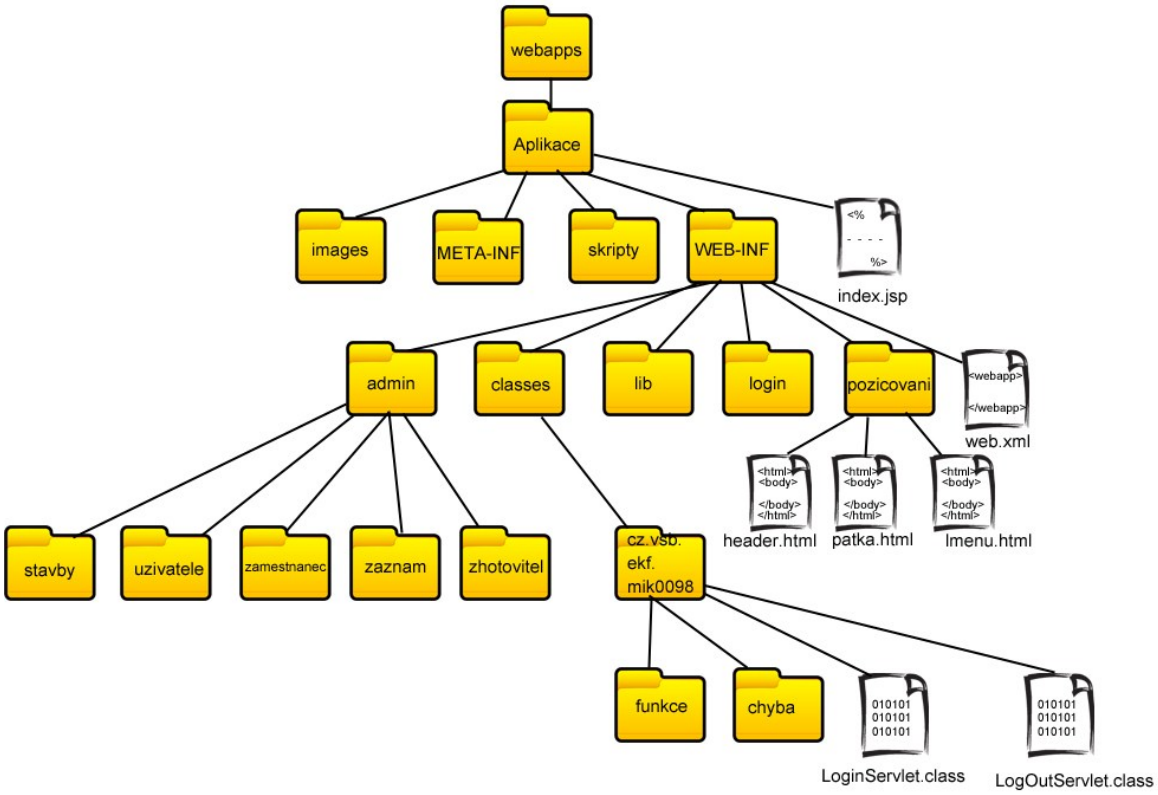
4.2.2.2 Struktura webové aplikace

Struktura webové aplikace je založena na modelu MVC (Model-View-Controler). Tento model odděluje prezentační logiku od aplikační. JSP stránky slouží pro zobrazení dat a Servlety pro práci s databází, jako jsou mazání, aktualizace a vložení nových záznamů v databázi.



Obrázek 4-5 MVC model

Struktura složek, která je zobrazena na obrázku 4-6. Slouží pro lepší pochopení celkové webové aplikace. Jelikož webová aplikace obsahuje několik stovek řádků kódu, 31 JSP stránek, 6 Java Servletů a 3 HTML stránky je pro přehlednost práce zmíněno jen několik z nich.



Obrázek 4-6 Struktura složek webové aplikace

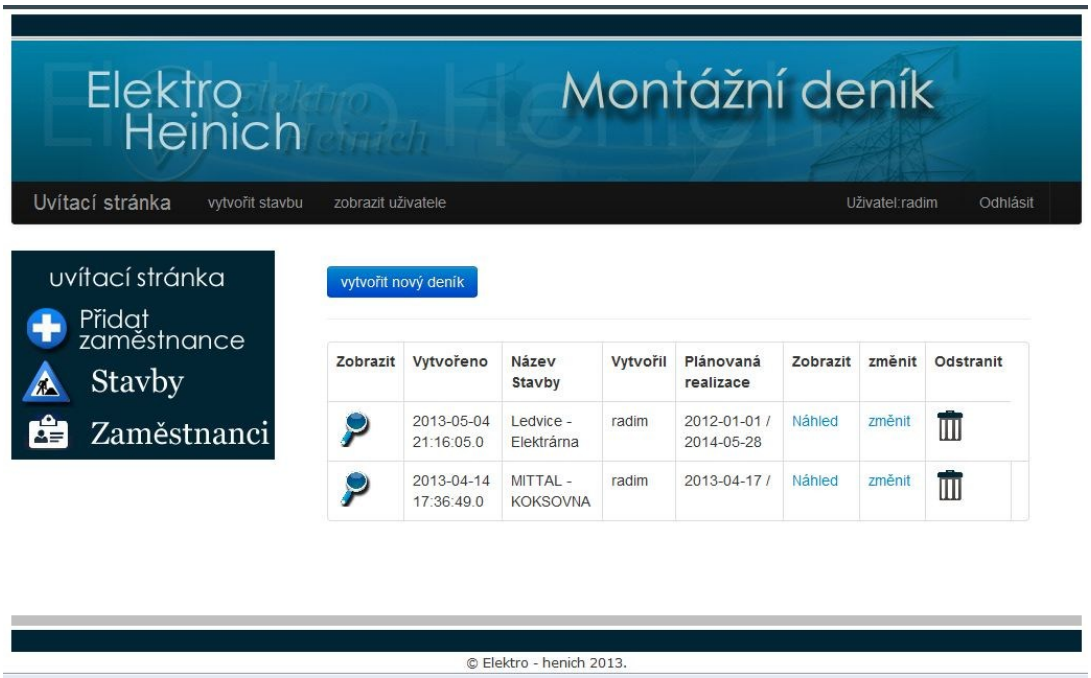
Jak vidíme na obrázku 4-6. Webové aplikace se typicky skládá z hierarchické struktury složek. Kořenová složka slouží jako kořen dokumentu pro webovou aplikaci v našem případě je pojmenována Aplikace. Pod kořenem se nachází několik složek, nejpodstatnější složka je WEB-INF, kde umísťujeme soubory, ke kterým chceme mít zabezpečený přístup. V této složce je umístěn i deskriptor (web.xml). Deskriptor je soubor, ve kterém je umístěna konfigurace webové aplikace. Další složka je WEB-INF/classes, která je použita pro Java Servlety a třídy, které jsou použity ve webové aplikaci. Složka /classes musí obsahovat řádný popis podsložek, přesně podle jména balíčku. Druhou složkou je WEB-INF/lib, která obsahuje knihovny, které jsou v našem případě Connector/J a JSTL 1.1. Musíme poznamenat, že veškerou práci ve webové aplikaci řídí Java Servlety, které přidávají uživateli oprávnění. Tato oprávnění specifikují uživateli práva, které má v rámci dané aplikace. Podrobnější popis Java Servletů a jejich vazby s tabulkami naleznete příloze č. 5.

4.2.2.3 Uživatelské oprávnění

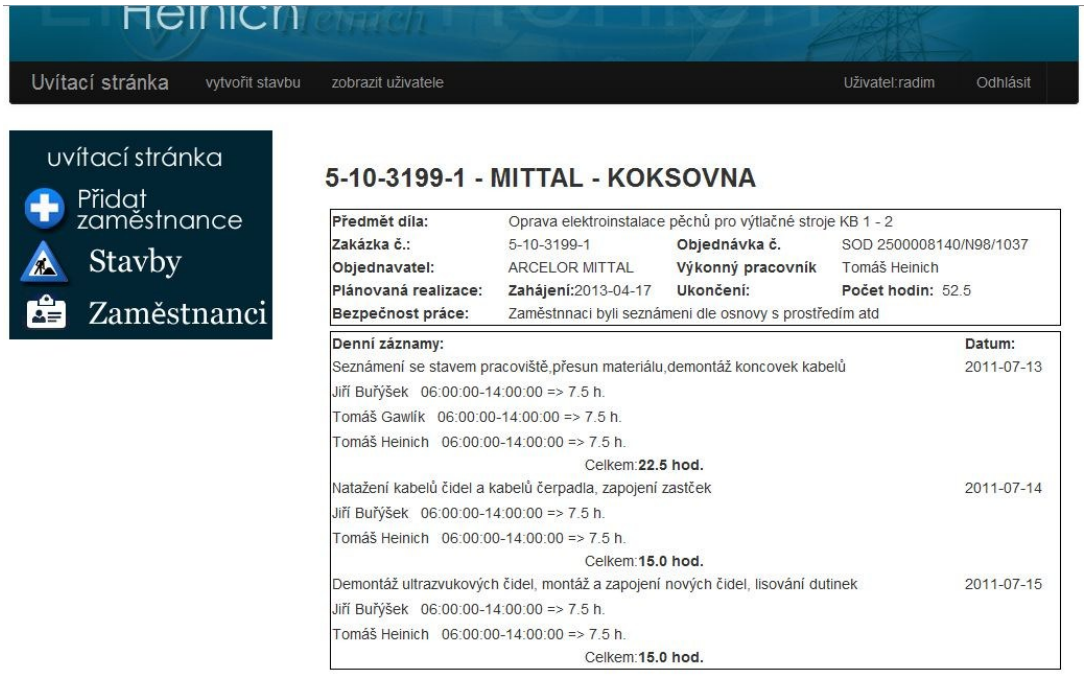
Pro přihlášení do informačního systému Montážního deníku slouží Servlet s názvem LoginServlet.class. Tento Servlet pracuje s parametry, které získá z metody post z protokolu HTTP. Dále porovná hodnoty z webového formuláře, které jsou přihlašovací jméno a heslo, jestli se nachází v tabulce uživatelů. Pokud má uživatel roli „admin“ nebo „ceo“ jeho požadavek se přesměruje na stránku WEB-INF/admin/adminPage.jsp, které slouží administrátorské rozhraní, ale jestli je jeho role „member“ dostane se na stránku WEB-INF/login/success.jsp, které poskytuje rozhraní pro zaměstnance firmy.

Administrátorské rozhraní

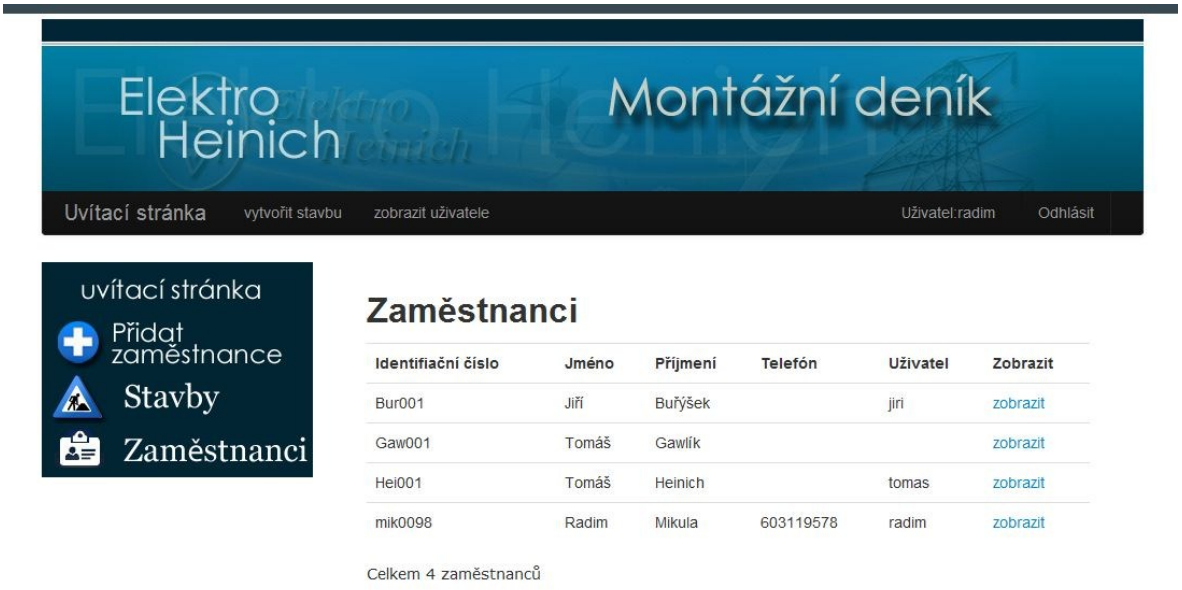
Administrátor vykonává správu celé webové aplikace. Jeho hlavní funkcí je obsluha uživatelských účtů. Má přehled o všech zakázkách a může je také upravovat, jak můžeme vidět na obrázku 4-7. Po kliknutí na danou zakázku zobrazuje stránka celkový přehled zakázky, který má podobu papírového Montážního deníku, ukázku této funkce vidíme na obrázku 4-8. Dále má přehled o zaměstnancích na jednotlivých zakázkách, tuto funkci vidíme na obrázku 4-9. Druhou hlavní roli webové aplikace je výkonný pracovník. Označovaný v databázovém systému jako „coe“ (Chief Executive Officer, výkonný pracovník). Tento uživatel se po přihlášení dostane na stejnou stránku jako administrátor, ale má poměrně menší možnosti v dané aplikaci. Může spravovat jen zakázky, ve kterých je výkonný pracovník nebo ty zakázky, které sám vytvořil. Dále může povolit určitým zaměstnancům zapisovat hodiny do přesně specifikované zakázky. Oproti administrátorovi nemůže zakázky mazat, a také nemá přístup k informacím o zaměstnanci.



Obrázek 4-7 Přehled zakázek



Obrázek 4-8 Celkový přehled zakázky

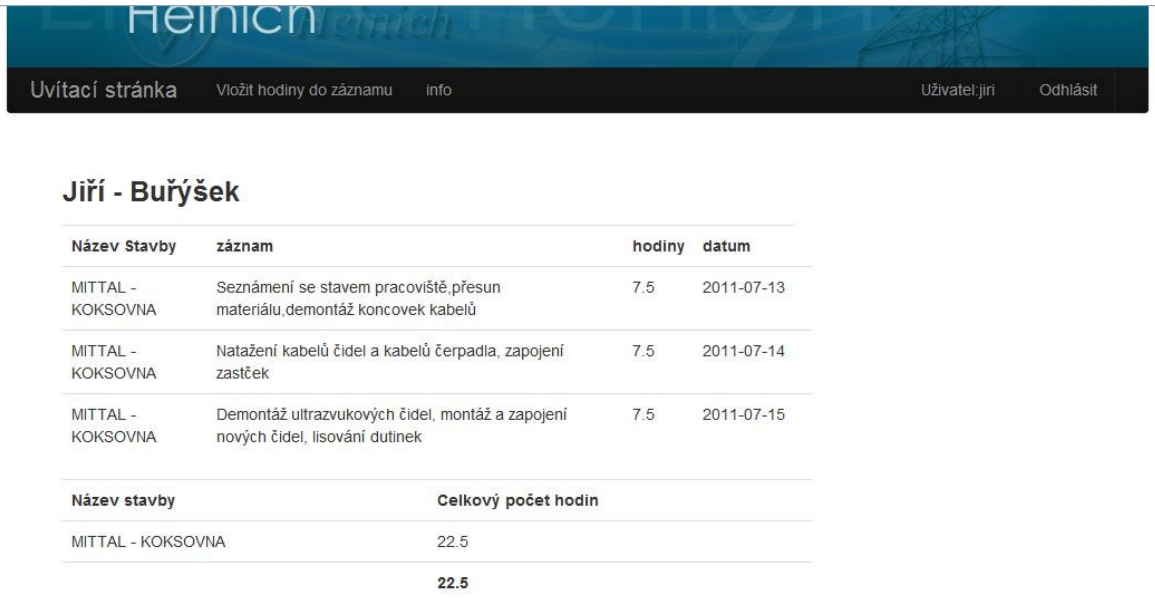


Obrázek 4-9 Zaměstnanci

Rozhraní pro zaměstnance

Zaměstnanec je v aplikaci označený jako „member“ (člen), po přihlášení se dostává na jinou stránku než administrátor. Jeho hlavní funkce jsou vkládání hodin k jednotlivým záznamům v zakázce, ke které má přístup. Tuto funkci zobrazuje obrázek 4-10. Dále je mu poskytnut přehled o všech svých činností se součtem hodin v zakázkách, na kterých se podílel viz obrázek 4-11.

Obrázek 4-10 Přehled zakázek daného uživatel



Obrázek 4-11 Přehled činností a hodin zaměstnance

4.2.3 Bezpečnost

Bezpečnost systému je relativně široká disciplína. V této práci zmiňujeme pouze základní vlastnosti v rámci bezpečnosti informačních systému:

- důvěrnost dat – vlastnost dat, která znemožňuje odhalení neautorizovanými jedinci,
- dostupnost dat – zajišťuje, aby data byla k dispozici na vyžádání autorizované osoby,
- integrita dat – vlastnost dat, která nebyla změněna nebo zničena neautorizovaným způsobem

Výše uvedené vlastnosti byly v rámci webové aplikace zajištěný pomocí přihlašování do systému. Díky vlastnosti role, kterou přidáváme ke každému uživateli, zabezpečujeme autorizovaný přístup k datům. Každý uživatel má i své heslo.

Pro zajištění bezpečnosti aplikace se ukládá místo hesel do databáze jejich hashované tvary pomocí algoritmu SHA-1. Hash algoritmus vytváří hashovaný tvar, obvykle 128-bitové hodnoty, který je výsledkem jednosměrné matematické kalkulace (hash algoritmu), jež zabezpečí, že není možné se obráceným postupem dobrat k původní hodnotě. Proto se při přihlašování porovnávají hashované řetězce.

5 Zhodnocení navržené koncepce a její implementace na portálu firmy

Základem celé realizace systému je detailní analýza vycházející ze stanovených požadavků. Z analýzy a rozboru technologií vychází návrh jednotlivých částí systému. Ten je postaven na architektuře klient – server, přesněji její variantě - třívrstvém modelu. Třívrstvý model slouží pro efektivnější vytváření distribuovaných systémů. Jeho důsledné oddělení prezentační a aplikační logiky přináší výhody v návrhu celého systému. Prezentační logika nemusí řešit složité procesy a může se soustředit na zobrazování výsledků a volání aplikační logiky. Aplikační logika umožňuje manipulaci s daty v databázi a rozlišovat oprávnění, které mají jednotliví uživatelé. Perzistentní uchování dat, ale také základní zajištění integritních omezení a konzistence. Plní v třívrstvé architektuře databázový systém.

Navržený systém splňuje všechny základní požadavky, které vytvářejí dobrý základ systému a dávají prostor pro budoucí vylepšení. Již nyní existují návrhy na další funkce, které by systém mohl nabízet.

Jmenujeme alespoň některé:

- diskuze na webu,
- export informací do různých formátů,
- rozlišení datumů na víkendy pro příplatky zaměstnanců.

5.1 Implementace aplikace na portál firmy

Jelikož webhosting firmy, což je pronájem prostoru pro webové stránky na cizím serveru, neposkytuje technologie, kterou využívá navržený systém. Z uvedeného důvodu se implementace webové aplikace odkládá. Proto se implementace webové aplikace neuskuteční v termínu odevzdání této bakalářské práce. Firma ale zvažuje zakoupení nového webhostingu a to od společnosti Warnet.cz s. r. o., která nabízí telekomunikační služby a od roku 2007 webhostingové služby pod značkou hiweb.cz., kde dále naleznete jejich ceník za měsíční služby.

6 Závěr

Jádrem celé práce bylo studovat klíčové Java technologie JSP, Java Servlety a prakticky je použít ve webové aplikaci systému Montážního deníku, která slouží jako informační systém dané firmy a plní funkce elektronického Montážního deníku.

Druhý cíl byl analýza montážního deníku a navržení webové aplikace. Díky spolupráci s vedením firmy byl naplněn i tento cíl. Uvedená analýza se stala dobrým podkladem pro konceptuální model a jeho následná realizace do relačního modelu, který je spjatý s MySQL databázovým systémem.

Třetím cílem byl vytvořit konceptuální model. Tento model sloužil pro lepší pochopení datové struktury a logickou propojenost tabulek, které obsahují veškeré informace, společně s cizími klíči. Díky konceptuálního modelu bylo možné převést tento model do databázového systému MySQL a vytvořit tak relační datový model. Další výhodou modelu je přehlednost a usnadnění rozšířenosti infomačního systému.

Posledním cílem byla realizace webového rozhraní, která byla zajištěna pomocí zmiňované technologie. Podrobné zhodnocení i náměty pro možné rozšíření aplikace jsou uvedeny v předchozí kapitole 5. Navržený systém oproti papírovému deníku nabízí mnoho výhod. Hlavní přínos spočívá v automatizaci procesu, dostupnosti a přehlednosti všech zakázek. Další vývoj navrženého systému může být např. rozlišení hodinových sazeb, které se konají v noci a o víkendu. Takový systém by sloužil pro veškeré firmy, které rozlišují hodinový sazebník.

Nezbytnou přílohou této práce je CD, které kromě textu obsahuje zdrojové kódy i jejich zkompilovanou formu. Kód je pečlivě členěn do jednotlivých balíčků podle účelu použití.

Nejpodstatnějším osobním přínosem práce je získaní detailních informací z široké oblasti návrhu softwaru a programování na Java platformě. Pevně doufám, že systém Montážního deníku nabídne podniku důmyslný nástroj, který povede zefektivnění práce interních zaměstnanců. V budoucnu snad bude systém sloužit pro všechny zakázky dané firmy.

Seznam použité literatury

[1] BASHAM, Bryan, Kathy SIERRA a Bert BATES. Head first servlets and JSP. 2nd ed. Sebastopol, [Calif.]: O'Reilly, c2008, xxxii, 879 p. ISBN 978-059-6516-680.

[2] CADENHEAD, Rogers, Kathy SIERRA a Bert BATES. Sams teach yourself Java in 24 hours. 6th ed. Indianapolis, Ind.: Sams, c2012, vi, 419 p. ISBN 06-723-3575-1.

[3] HALL, Marty, Kathy SIERRA a Bert BATES. Java: servlety a stránky JSP. 6th ed. Praha: Neocortex, 2001, xviii, 585 s. ISBN 80-863-3006-0

[4] HEFFELFINGER, David R. Java EE 6 development with Netbeans 7: develop professional enterprise Java EE applications quickly and easily with this popular IDE. Birmingham, U.K.: Packt Open Source, 2011, iv, 374 p. Community experience distilled. ISBN 978-1-849512-70-1.

[5] HEROUT, Pavel. Učebnice jazyka Java. 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.

[6] JENDROCK, Eric. The Java EE 6 tutorial: basic concepts. 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2011, xxviii, 557 p. ISBN 01-370-8185-5.

[7] KALUŽA, Jindřich a Ludmila KALUŽOVÁ. Modelování dat v informačních systémech. 1. vyd. Praha: Ekopress, 2012, 125 s. ISBN 978-80-86929-81-1.

[8] KOFLER, Michael. Mistrovství v MySQL 5. Vyd. 1. Překlad Jan Svoboda, Ondřej Baše, Jaroslav Černý. Brno: Computer Press, 2007, 805 s. ISBN 978-80-251-1502-2.

[9] SCHILDT, Herbert. Java 7: výukový kurz. 1. vyd. Brno: Computer Press, 2012, 664 s. ISBN 978-80-251-3748-2.

Seznam zkratek

API	Application Programming Interface
atd.	a tak dále
CSS	Cascading Style Sheets
č.	číslo
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
Java EE	Java Enterprise Edition
Java ME	Java Micro Edition
Java SE	Java Standard Edition
JDBC	Java Database Connectivity
JDK	Java Development Kit
JSF	JavaServer Faces
JSP	Java Server Pages
JSTL	JavaServer Pages Standard Tag Library
JVM	Java Virtual Machine
MVC	Model-view-controller
např.	například
popř.	popřípadě
SQL	Structured Query Language
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
XML	Extensible Markup Language

Seznam tabulek

Seznam obrázků

Prohlášení o využití výsledků bakalářské práce

Prohlašuji, že

- jsem byl seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;

- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, bakalářskou práci užít (§ 35 odst. 3);

- souhlasím s tím, že bakalářská práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího bakalářské práce. Souhlasím s tím, že bibliografické údaje o bakalářské práci budou zveřejněny v informačním systému VŠB-TUO;

- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;

- bylo sjednáno, že užít své dílo, bakalářskou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Havířově, dne 3. 5. 2013

Radim Mikula

Adresa trvalého pobytu studenta:

Obránců Míru 829/4

736 01 Havířov-Šumbark

Seznam příloh

Příloha č. 1 Formulář Montážního deníku

Montážní deník

HEINICH elektro

list č.: 005038

List ________ z ________ listů

Předmět díla: ....

....

Zakázka č.: .... Objednávka č.: ....

Objednatel: .... Zhotovitel: ....

Oprávněný pracovník: .... Oprávněný pracovník: ....

Plánovaná realizace: Zahájení: .... Ukončení: .... Počet hodin: ....

Související dokumentace: ....

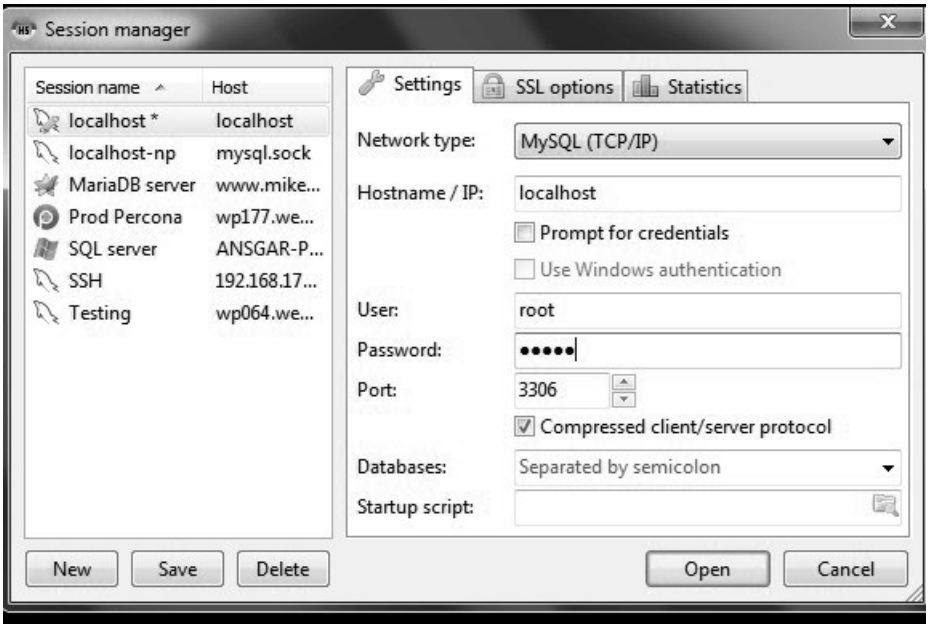
Bezpečnost práce:	Datum:
Denní záznamy:	

Převzetí díla objednatelem (podpis oprávněného prac.):	Předání díla zhotovitelem (podpis oprávněného prac.):	

Obrázek: Formulář Montážního deníku

Příloha č. 2 Připojení k databázi v operačním systému Windows

Prvním krokem pro vytvoření naší databáze je navázat spojení s databázovým serverem Ze všeho nejdříve se objeví dialogové okno, které žádá o název hostitele a pověření. Seznam hledaných databází může být omezen a řazen podle abecedy. Na obrázku 4-4. můžeme vidět první dialogové okno.



Obrázek: Připojení k databázi

Po připojení k MySQL serveru je vhodné vytvořit nového uživatele, který má práva jen k databázi, pro tu kterou spravuje. Pro pohodlnější manipulaci byla nově vytvořenému uživateli přidělaná veškerá administrátorská práva.

```
GRANT ALL PRIVILEGES ON denik.* TO student@'%' IDENTIFIED BY 'student' WITH
GRANT OPTION;
```

Použití příkazu GRANT je způsob vytvoření uživatele, který specifikuje, ke které databázi má uživatel práva. Příkaz denik.* znamená na databázi deník a hvězdička všechny její tabulky. Příkaz student@'%' označuje uživatelské jméno student a znak % umožňuje připojení k databázi z libovolného počítače. Nakonec IDENTIFIED BY 'student' znamená heslo uživatele.

Vytvoření tabulek

Pro tvorbu tabulek slouží jazyk DDL, jsou příkazy jazyka SQL, které jsou určeny pro definici dat. MySQL podporuje DDL příkaz pro vytvoření tabulky CREATE TABLE. Při vytváření tabulek musíme zadat název, kterým se na ni budeme odkazovat v dalších SQL příkazech, a hlavně názvy sloupců, které bude tabulka obsahovat. U každého sloupce musíme zadat datový typ hodnot, které bude obsahovat – číslo, řetězec znaků, datum apod.

Tabulka stavby

```
CREATE TABLE IF NOT EXISTS `stavby` (
`cZakazky` char(13) COLLATE utf8_czech_ci NOT NULL,
`nazevStavby` varchar(30) COLLATE utf8_czech_ci NOT NULL,
`datumOd` date DEFAULT NULL,
`datumDo` date DEFAULT NULL,
`vytvoril` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`vytvoreno` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
PRIMARY KEY (`cZakazky`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci
```

Jak můžeme vidět u atributů s textovými řetězci (char, varchar), lze použít volitelný atribut CHARACTER SET název znakové sady COLLATE porovnání. Znakové sady určují, jaký kód označuje daný znak. UTF8 je převodní formát pro Unicode. Unicode je dvoubajtová znaková sada, která může obsahovat až 65 536 různých znaků, což stačí pro všechny evropské jazyky, ale i pro některé asijské jazyky. TIMESTAMP je datový typ, který je automaticky aktualizovaný, kdykoliv změníme daný záznam, tudíž uchovává časový údaj poslední změny záznamu. Primární klíč tabulky se zadává pomocí klíčového slova PRIMARY KEY. A pomocí ENGINE můžeme specifikovat typ tabulky.

Tabulka zakázky

```
DROP TABLE IF EXISTS `zakazka`;
CREATE TABLE IF NOT EXISTS `zakazka` (
`cZakazky` char(13) COLLATE utf8_czech_ci NOT NULL,
`PredmetDila` text COLLATE utf8_czech_ci NOT NULL,
`smlouva` varchar(50) COLLATE utf8_czech_ci NOT NULL,
`vykonnyPrac` varchar(50) COLLATE utf8_czech_ci DEFAULT NULL,
`bezpecnostPrace` text COLLATE utf8_czech_ci,
`Objednavatel` varchar(30) COLLATE utf8_czech_ci DEFAULT NULL,
`OpravnenyPrac` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`zahajeni` date DEFAULT NULL,
`ukonceni` date DEFAULT NULL,
PRIMARY KEY (`cZakazky`),
CONSTRAINT `zakazka_ibfk_1` FOREIGN KEY (`cZakazky`) REFERENCES `stavby`
(`cZakazky`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

Díky následující ukázky kódu byl nastaven cizí klíč. Pomocí klíčového slova FOREIGN KEY můžeme specifikovat vztahy mezi tabulkami. V tomto případě máme vztah 1: 1 díky tomu, že náš primární klíč je zároveň i cizí klíč.

Uložené procedury

Do kódu uložené procedury můžeme umístit téměř všechny příkazy či funkce jazyka SQL. Následující procedura slouží k počtu hodin, které ukládá do atributu datového typu FLOAT.

```
DELIMITER //
CREATE PROCEDURE `vlozitZhotovitele`(IN `idZho` int, IN `casOd` time, IN
`casDo` time, IN `prestavkaV` float, IN `idZam` char(10)
)
begin
insert into zhotovitel (idZhotovitele,cas_od,cas_do,
prestavka,idZamestnance)
values (idZho,casOd,casDo, prestavkaV, idZam);

update zhotovitel ch
join ( select idZamestnance, prestavka,((sum(hour(timediff(cas_do,cas_od)))
+ sum(minute(timediff(cas_do,cas_od))/60))-prestavka) hodinyL
from zhotovitel
where idZhotovitele=idZho and idZamestnance=idZam
)  v
on     ch.idZamestnance = v.idZamestnance
set    ch.hodiny = v.hodinyL
where  ch.idZhotovitele=idZho;
end//
DELIMITER ;
```

DELIMITER slouží pro změnu oddělovače u psaní procedur.

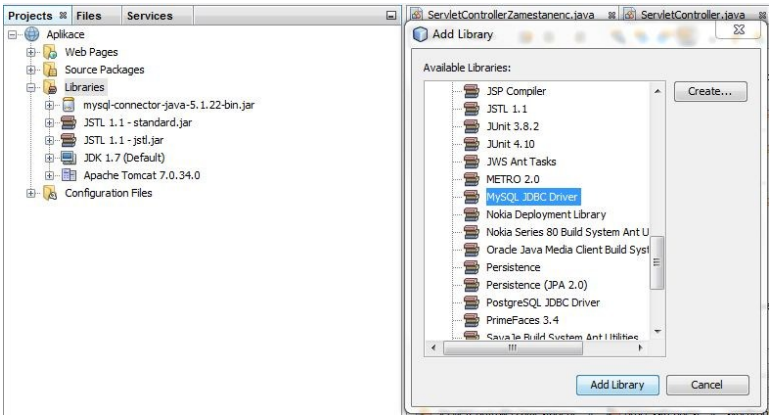
Zbytek zdrojových kódů databázového systému MySQL naleznete v příloze číslo 4.

Příloha č. 3 Připojení databáze v Netbeans IDE 7.3

Pro možnost práce s databází v naší webové aplikaci je nutné připojit databázi pomocí určitého driveru, který zajistí spojení. V popisovaném webovém rozhraní je použit driver Connector/J, což je v podstatě ovladač JDBC pro MySQL. JDBC je kolekce tříd pro programování databázových aplikací v jazyku Java. JDBC není závislé na žádném konkrétním databázovém systému.

Instalace

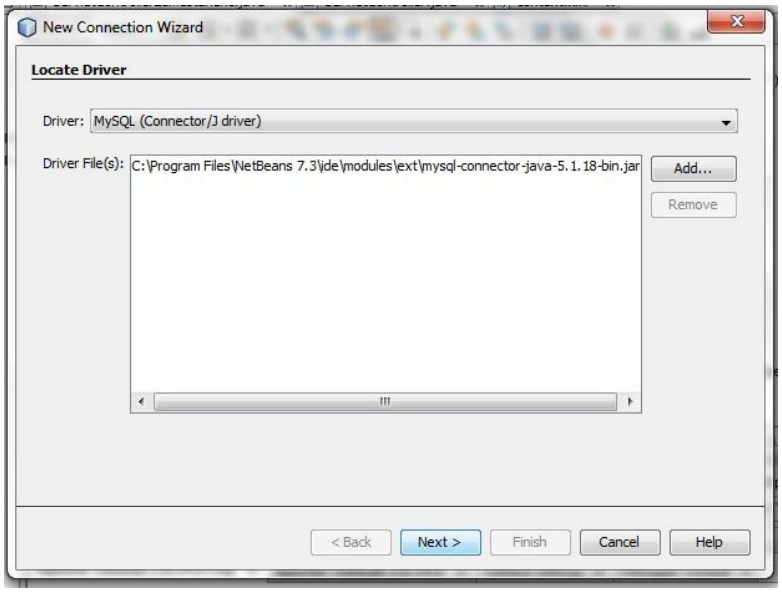
Connector/J je volně ke stažení na adrese http://www.mysql.com/ a v archivním formátu *.zip pro Windows. Instalace se skládá z rozbalení obsahu archivního souboru do zvolené složky. Kriticky důležitým bodem je, že běhové prostředí Javy musí umět při spouštění programu novou knihovnu najít. Nejjednodušší způsob je importovat do adresáře Libraries, pomocí pravého tlačítka myši lze přidat ovládač.



Obrázek: Vložení ovládače

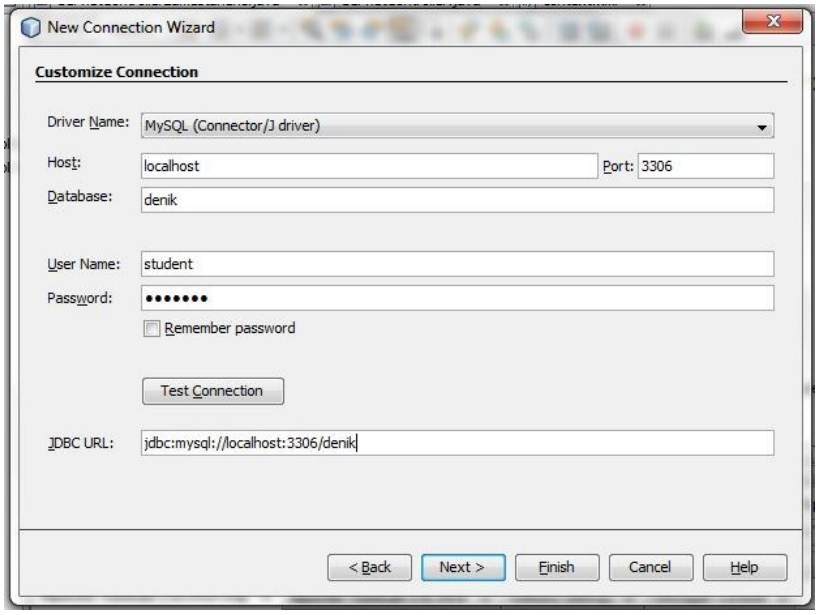
Vytvoření připojení

Po instalaci je nutné vytvořit připojení na kartě Services kliknutím pravým tlačítkem myši na ikonu Databases a vybrat New Connection. Po tomto kroku se zobrazí dialogové okno (viz obrázek: Výběr JDBC ovladače), kde je vybrán příslušný soubor s ovládačem ze složky.



Obrázek: Výběr JDBC ovladače

Po stisknutí tlačítka Next se nastavují pokročilé specifikace spojení. Viz obrázek Nastavení spojení.

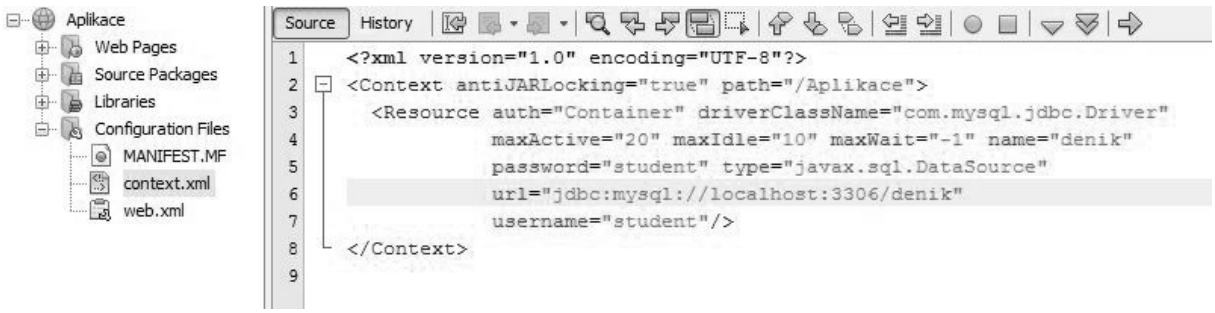


Obrázek: Nastavení spojení

Sdílení spojení

Posledním krokem pro práci s databází v našem projektu je vytvoření Sdílení spojení (Connection pooling), které bylo zmíněno v kapitole 2.2.10**.** Tuto akci za nás udělá vývojové prostředí NetBeans 7.3 automaticky. Stačí pouze v libovolném Java Servletu zmačknout levý

„alt + insert“ a klinout na „use database“ a vybrat námi vytvořené spojení. Po tomto kroku máme vytvořené sdílení spojení a jeho specifikace najdeme v souboru context.xml, které můžete vidět na obrázku Specifikace sdílení spojení.



Obrázek: Specifikace sdílení spojení

Toto byl poslední krok vytvoření spojení s databází, nyní je možno v prostředí NetBeans 7.3 manipulovat s daty v databázi.

Příloha č. 4 Zdrojové kódy databázového systému MySQL

```
DROP DATABASE IF EXISTS `denik`;
CREATE DATABASE IF NOT EXISTS `denik` /*!40100 DEFAULT CHARACTER SET utf8
COLLATE utf8_czech_ci */;
USE `denik`;


-- Dumping structure for procedure denik.pridatZamestnance
DROP PROCEDURE IF EXISTS `pridatZamestnance`;
DELIMITER //
CREATE DEFINER=`student`@`%` PROCEDURE `pridatZamestnance`(IN `Username`
varchar(20), IN `Passwrd` varchar(50), IN `Role` varchar(20), IN `Id`
CHAR(10))
begin
insert into uzivatele(username,passwrd,role,idZamestnance) values
(Username,Passwrd,Role,Id);

end//
DELIMITER ;


-- Dumping structure for table denik.stavby
DROP TABLE IF EXISTS `stavby`;
CREATE TABLE IF NOT EXISTS `stavby` (
`cZakazky` char(13) COLLATE utf8_czech_ci NOT NULL,
`nazevStavby` varchar(30) COLLATE utf8_czech_ci NOT NULL,
`datumOd` date DEFAULT NULL,
`datumDo` date DEFAULT NULL,
`vytvoril` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`vytvoreno` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
PRIMARY KEY (`cZakazky`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

-- Dumping data for table denik.stavby: ~2 rows (approximately)
/*!40000 ALTER TABLE `stavby` DISABLE KEYS */;
INSERT INTO `stavby` (`cZakazky`, `nazevStavby`, `datumOd`, `datumDo`,
`vytvoril`, `vytvoreno`) VALUES
('2932-20482', 'Ledvice - Elektrárna', '2012-01-01', '2014-05-28', 'radim',
'2013-05-04 21:16:05'),
('5-10-3199-1', 'MITTAL - KOKSOVNA', '2013-04-17', NULL, 'radim', '2013-04-
14 17:36:49');
/*!40000 ALTER TABLE `stavby` ENABLE KEYS */;


-- Dumping structure for table denik.uzivatele
DROP TABLE IF EXISTS `uzivatele`;
CREATE TABLE IF NOT EXISTS `uzivatele` (
`username` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`passwrd` varchar(50) COLLATE utf8_czech_ci NOT NULL,
`role` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`idZamestnance` char(10) COLLATE utf8_czech_ci DEFAULT NULL,
PRIMARY KEY (`username`),
KEY `idZamestnance` (`idZamestnance`),
CONSTRAINT `FK_uzivatele_zamestnanec` FOREIGN KEY (`idZamestnance`)
REFERENCES `zamestnanec` (`idZamestnance`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

```
-- Dumping data for table denik.uzivatele: ~3 rows (approximately)
/*!40000 ALTER TABLE `uzivatele` DISABLE KEYS */;
INSERT INTO `uzivatele` (`username`, `passwrd`, `role`, `idZamestnance`)
VALUES
('jiri', '1b548d10035ae428195c41f5d55eedb1437ccd8a', 'member', 'Bur001'),
('radim', '2212ec59ea1d2c2f299a3c7cb060cb84b52d4222', 'admin', 'mik0098'),
('tomas', '2bc6038c3dfca09b2da23c8b6da8ba884dc2dcc2', 'coe', 'Hei001');
/*!40000 ALTER TABLE `uzivatele` ENABLE KEYS */;


-- Dumping structure for procedure denik.vlozitZhotovitele
DROP PROCEDURE IF EXISTS `vlozitZhotovitele`;
DELIMITER //
CREATE DEFINER=`student`@`%` PROCEDURE `vlozitZhotovitele`(IN `idZho` int,
IN `casOd` time, IN `casDo` time, IN `prestavkaV` float, IN `idZam`
char(10)
)
begin
insert into zhotovitel (idZhotovitele,cas_od,cas_do,
prestavka,idZamestnance)
values (idZho,casOd,casDo, prestavkaV, idZam);

update zhotovitel ch
join ( select idZamestnance, prestavka,((sum(hour(timediff(cas_do,cas_od)))
+ sum(minute(timediff(cas_do,cas_od))/60))-prestavka) hodinyL
from zhotovitel
where idZhotovitele=idZho and idZamestnance=idZam
)  v
on     ch.idZamestnance = v.idZamestnance
set    ch.hodiny = v.hodinyL
where  ch.idZhotovitele=idZho;


end//
DELIMITER ;


-- Dumping structure for table denik.zakazka
DROP TABLE IF EXISTS `zakazka`;
CREATE TABLE IF NOT EXISTS `zakazka` (
`cZakazky` char(13) COLLATE utf8_czech_ci NOT NULL,
`PredmetDila` text COLLATE utf8_czech_ci NOT NULL,
`smlouva` varchar(50) COLLATE utf8_czech_ci NOT NULL,
`vykonnyPrac` varchar(50) COLLATE utf8_czech_ci DEFAULT NULL,
`bezpecnostPrace` text COLLATE utf8_czech_ci,
`Objednavatel` varchar(30) COLLATE utf8_czech_ci DEFAULT NULL,
`OpravnenyPrac` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`zahajeni` date DEFAULT NULL,
`ukonceni` date DEFAULT NULL,
PRIMARY KEY (`cZakazky`),
CONSTRAINT `zakazka_ibfk_1` FOREIGN KEY (`cZakazky`) REFERENCES `stavby`
(`cZakazky`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

-- Dumping data for table denik.zakazka: ~2 rows (approximately)
/*!40000 ALTER TABLE `zakazka` DISABLE KEYS */;
INSERT INTO `zakazka` (`cZakazky`, `PredmetDila`, `smlouva`, `vykonnyPrac`,
`bezpecnostPrace`, `Objednavatel`, `OpravnenyPrac`, `zahajeni`, `ukonceni`)
VALUES
```

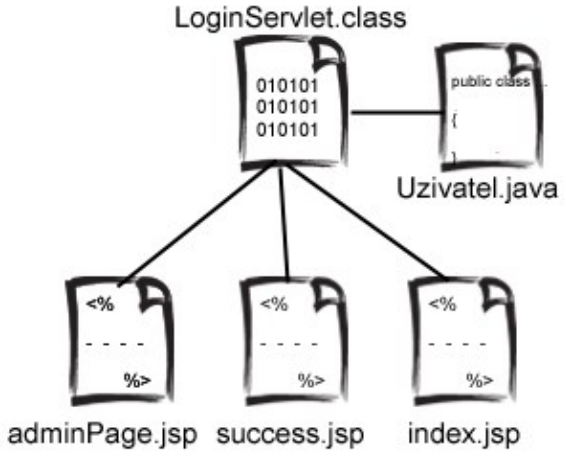
```
('2932-20482', 'Zabudování EPS', '20830832', 'Hei001', '', 'SINIT', 'ing.
Tkáč', NULL, NULL),
('5-10-3199-1', 'Oprava elektroinstalace pěchů pro výtlačné stroje KB 1 -
2', 'SOD 2500008140/N98/1037', 'Hei001', '    Zaměstnnaci byli seznámeni
dle osnovy s prostředím atd\r\n    \r\n    ', 'ARCELOR MITTAL', 'Ing. Tkáč
P.', NULL, NULL);
/*!40000 ALTER TABLE `zakazka` ENABLE KEYS */;


-- Dumping structure for table denik.zamestnanec
DROP TABLE IF EXISTS `zamestnanec`;
CREATE TABLE IF NOT EXISTS `zamestnanec` (
`idZamestnance` char(10) COLLATE utf8_czech_ci NOT NULL,
`Jmeno` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`Prijmeni` varchar(20) COLLATE utf8_czech_ci NOT NULL,
`telefon` int(11) DEFAULT NULL,
PRIMARY KEY (`idZamestnance`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

-- Dumping data for table denik.zamestnanec: ~4 rows (approximately)
/*!40000 ALTER TABLE `zamestnanec` DISABLE KEYS */;
INSERT INTO `zamestnanec` (`idZamestnance`, `Jmeno`, `Prijmeni`, `telefon`)
VALUES
('Bur001', 'Jiří', 'Buřýšek', NULL),
('Gaw001', 'Tomáš', 'Gawlík', NULL),
('Hei001', 'Tomáš', 'Heinich', NULL),
('mik0098', 'Radim', 'Mikula', 603119578);
/*!40000 ALTER TABLE `zamestnanec` ENABLE KEYS */;


-- Dumping structure for table denik.zaznamy
DROP TABLE IF EXISTS `zaznamy`;
CREATE TABLE IF NOT EXISTS `zaznamy` (
`cZakazky` char(13) COLLATE utf8_czech_ci NOT NULL,
`datum` date NOT NULL,
`zaznam` mediumtext COLLATE utf8_czech_ci NOT NULL,
`IdZhotovitele` int(6) NOT NULL AUTO_INCREMENT,
`vlozil` varchar(50) COLLATE utf8_czech_ci DEFAULT NULL,
PRIMARY KEY (`IdZhotovitele`),
KEY `cZakazky` (`cZakazky`),
CONSTRAINT `zaznamy_ibfk_1` FOREIGN KEY (`cZakazky`) REFERENCES `zakazka`
(`cZakazky`)
) ENGINE=InnoDB AUTO_INCREMENT=30 DEFAULT CHARSET=utf8
COLLATE=utf8_czech_ci;

-- Dumping data for table denik.zaznamy: ~3 rows (approximately)
/*!40000 ALTER TABLE `zaznamy` DISABLE KEYS */;
INSERT INTO `zaznamy` (`cZakazky`, `datum`, `zaznam`, `IdZhotovitele`,
`vlozil`) VALUES
('5-10-3199-1', '2011-07-13', 'Seznámení se stavem pracoviště,přesun
materiálu,demontáž koncovek kabelů', 1, NULL),
('5-10-3199-1', '2011-07-14', 'Natažení kabelů čidel a kabelů čerpadla,
zapojení zastček', 27, 'radim'),
('5-10-3199-1', '2011-07-15', 'Demontáž ultrazvukových čidel, montáž a
zapojení nových čidel, lisování dutinek', 29, 'radim');
/*!40000 ALTER TABLE `zaznamy` ENABLE KEYS */;


-- Dumping structure for table denik.zhotovitel
DROP TABLE IF EXISTS `zhotovitel`;
CREATE TABLE IF NOT EXISTS `zhotovitel` (
```

```
`IdZhotovitele` int(6) NOT NULL,
`cas_od` time NOT NULL,
`cas_do` time NOT NULL,
`prestavka` float DEFAULT NULL,
`hodiny` float DEFAULT NULL,
`idZamestnance` char(10) COLLATE utf8_czech_ci NOT NULL DEFAULT '',
KEY `FK_zhotovitel_zaznamy` (`IdZhotovitele`),
KEY `FK_zhotovitel_zamestnanec` (`idZamestnance`),
CONSTRAINT `FK_zhotovitel_zamestnanec` FOREIGN KEY (`idZamestnance`)
REFERENCES `zamestnanec` (`idZamestnance`),
CONSTRAINT `FK_zhotovitel_zaznamy` FOREIGN KEY (`IdZhotovitele`) REFERENCES
`zaznamy` (`IdZhotovitele`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;

-- Dumping data for table denik.zhotovitel: ~7 rows (approximately)
/*!40000 ALTER TABLE `zhotovitel` DISABLE KEYS */;
INSERT INTO `zhotovitel` (`IdZhotovitele`, `cas_od`, `cas_do`, `prestavka`,
`hodiny`, `idZamestnance`) VALUES
(1, '06:00:00', '14:00:00', 0.5, 7.5, 'Bur001'),
(1, '06:00:00', '14:00:00', 0.5, 7.5, 'Hei001'),
(1, '06:00:00', '14:00:00', 0.5, 7.5, 'Gaw001'),
(27, '06:00:00', '14:00:00', 0.5, 7.5, 'Bur001'),
(27, '06:00:00', '14:00:00', 0.5, 7.5, 'Hei001'),
(29, '06:00:00', '14:00:00', 0.5, 7.5, 'Bur001'),
(29, '06:00:00', '14:00:00', 0.5, 7.5, 'Hei001');
/*!40000 ALTER TABLE `zhotovitel` ENABLE KEYS */;


-- Dumping structure for procedure denik.zmenitZamestnance
DROP PROCEDURE IF EXISTS `zmenitZamestnance`;
DELIMITER //
CREATE DEFINER=`student`@`%` PROCEDURE `zmenitZamestnance`(IN `StareId`
VARCHAR(20), IN `NoveUsername` varchar(20), IN `NovePasswrd` varchar (50),
IN `StarePasswrd` varchar(50), IN `NoveRole` varchar(20))
begin
update uzivatele set username = NoveUsername, passwrd = NovePasswrd, role =
NoveRole
where username = StareId and passwrd = StarePasswrd;
end//
DELIMITER ;


-- Dumping structure for procedure denik.zmenitZhotovitele
DROP PROCEDURE IF EXISTS `zmenitZhotovitele`;
DELIMITER //
CREATE DEFINER=`student`@`%` PROCEDURE `zmenitZhotovitele`(IN `idZho` int,
IN `casOd` time, IN `casDo` time, IN `prestavkaV` float, IN `idZam`
char(10), IN `idZamNew` char(10)
)
begin
update zhotovitel set cas_od = casOd, cas_do = casDo, prestavka =
prestavkaV, idZamestnance = idZamNew
where idZhotovitele=idZho and idZamestnance = idZam;

update zhotovitel ch
join ( select idZamestnance, prestavka,((sum(hour(timediff(cas_do,cas_od)))
+ sum(minute(timediff(cas_do,cas_od))/60))-prestavka) hodinyL
from zhotovitel
where idZhotovitele=idZho and idZamestnance=idZamNew
)  v
on     ch.idZamestnance = v.idZamestnance
```

```
set     ch.hodiny = v.hodinyL
where   ch.idZhotovitele=idZho;


end//
DELIMITER ;
/*!40014 SET FOREIGN_KEY_CHECKS=1 */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

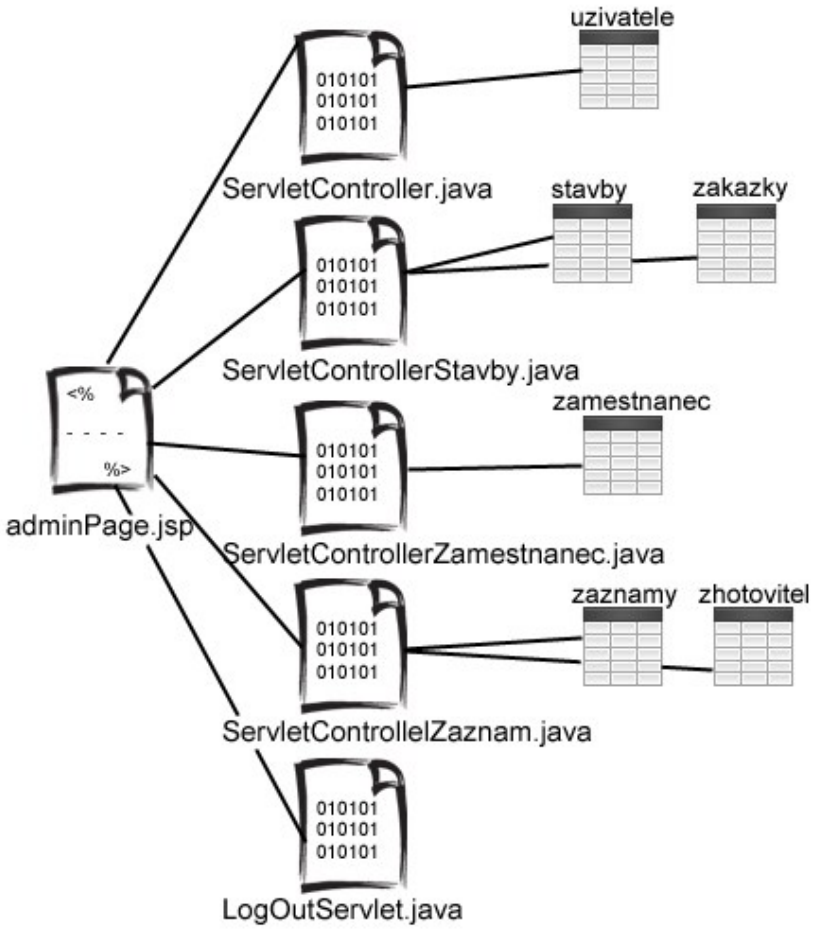
Příloha č. 5: Java Servlety a jejich vazby

Java Servlety slouží jako aplikační logika systému Montážní deník. Zabezpečují práci s databází a řídí celou aplikaci. V této příloze rozebereme všechny Java Servlety v aplikaci. První Java Servlet je LoginServlet slouží pro přihlášení do aplikace, porovnává přihlašovací údaje (jméno a heslo) v databázi v tabulce uživatele a přeposílá uživatele na stránku podle toho jakou má roli v daném systému. Pokud má roli „admin“ nebo „coe“ dostane se na stránku WEB-INF/admin/adminPage.jsp, jestli je uživatel „member“ Java Servlet ho přepošle na stránku WEBINF/login/success.jsp. V případě selhání přihlášení, přepošle uživatele zpátky na index.jsp. V případě úspěšného přihlášení daný Java Servlet zabezpečí vytvoření instance uživatele podle třídy Uzivatel.java. Tato třída má dvě datové položky typu String (uživatel a role).



Obrázek: LoginServlet

Pokud je uživatel na stránce adminPage.jsp může pracovat se všemi Java Servlety v aplikaci a využívat všechny jejich funkce, ale jen v případě, že uživatel je „admin“. Uživatel typu „coe“ má sice přístup ke všem Java Servletům, ale nemá oprávnění pro využití všech jejich funkcí. Jako je např. vložení nového zaměstnance, přehled všech zakázek nebo správa uživatelů. Dále nemůžu mazat libovolné záznamy v databázi. Na obrázku Java Servlety a jejich vazby nad tabulkami lze vidět Java Servlety, které jsou určené uživateli typu „admin“ nebo „coe“.



Obrázek: Java Servlety a jejich vazby nad tabulkami

ServletController slouží pro vytváření, mazání, popř. změnu uživatele. Přístup k danému Java Servletu má pouze uživatel, který má roli „admin“. Tato komponenta pracuje s tabulkou uživatele.

ServletControllerStavby zabezpečuje správu tabulek Stavby a Zakázky, a také zabezpečují jejich synchronizaci, protože pro příkazy jazyka SQL (INSERT, UPDATE, DELETE) jsou využity transakce, které pracují nad zmíněnými tabulkami. Transakce je konečná posloupnost operací, které jsou buďto všechny úspěšně realizovány a jejich výsledek je permanentně uložen v databázi, nebo nejsou úspěšně realizovány a systém navrátí databází do původního stavu před spuštěním transakce. Uvedené tabulky mají mezi sebou vztah 1: 1. Uživatel typu „coe“ nemůže provádět změny v záznamu stavby, ve kterých není vlastníkem nebo výkonným pracovníkem.

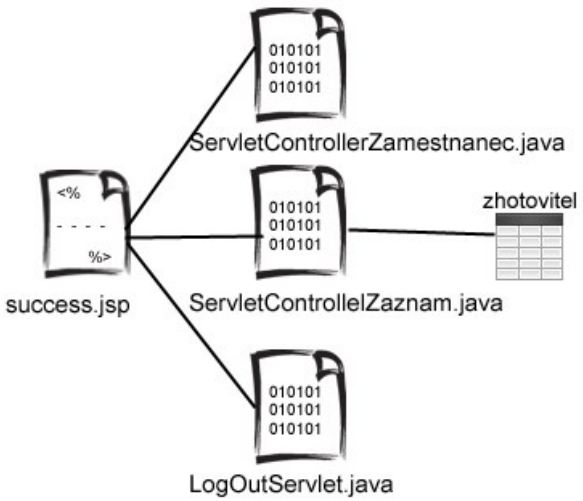
ServletControllerZamestnanec slouží pro práci s tabulkou zaměstnanec. Pro uživatele „admin“ zajišťuje celkovou správu tabulky, dále pro uživatele „admin“ a „coe“ umožňuje

přeposlání požadavku na stránku WEB-INF/admin/zamestnanec/zamestnanecPrehled.jsp, která slouží pro prezentaci celkového přehledu daného zaměstnance.

ServletControllerZaznam umožňuje provádět operaci nad tabulkami záznamy a zhotovitel. K tomuto Java Servletu mají přístup všichni uživatelé, ale jen „admin“ může provádět všechny jeho funkce. Uživatel „coe“ může k dané zakázce, pod kterou je vlastníkem nebo výkonným pracovníkem přidávat záznamy, ale také i zhotovitele, kteří daný záznam vykonávají.

LogOutServlet je přístupný pro všechny uživatele a je dostupný z jakékoliv JSP stránky. Tato komponenta slouží pro bezpečné odhlášení z webové aplikace.

Uživatel zaměstnanec označený v aplikaci jako „member“ má v systému minimální funkce. Po přihlášení na stránku success.jsp může přistupovat pouze k Java Servletům, které jsou ServletControllerZaznam a ServletControllerZamestnanec. Tyto komponenty zaměstnanci dovolují vykonávat jeho funkce, které jsou vložení hodin do záznamu v zakázce, ke které má oprávnění a možnost přeposlání na stránku WEB-INF/login/zamestnanecInfo.jsp pro zobrazení jeho celkového přehledu nad zakázkami. Na obrázku Java Servlety pro zaměstnance vidíme komponenty, ke kterým má přístup.



Obrázek: Java Servlety pro zaměstnance

Příloha č. 6: CD

Obsah CD:

1. Tento text bakalářské práce.
2. Zdrojové kódy, kompilace i deskriptory informačního systému Montážního deníku pro Java EE server Apache Tomcat 7.0
3. Základní struktura navržené databáze ve formě SQL příkazů. Dále minimálním množstvím dat předvyplněných pro MySQL databázi.